

RSP Test Specification for the eUICC

SGP.23-1 Version 3.1.2

26 April 2024

Security Classification: Non-Confidential

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

Copyright Notice

Copyright © 2024 GSM Association

Disclaimer

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

Antitrust Notice

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

**Table of Contents**

1 Introduction 5

1.1 Overview 5

1.2 Scope 5

1.3 Definition of Terms 5

1.4 Abbreviations 7

1.5 Document Cross-references 8

1.6 Conventions 10

2 Testing Rules 10

2.1 Applicability 10

2.1.1 Format of the Optional Features Table 10

2.1.2 Format of the Applicability Table 10

2.1.3 Applicability and Notations 11

2.1.4 Optional Features Table 11

2.1.5 Applicability Table 12

2.2 General Consideration 21

2.2.1 Test Case Definition 21

2.2.2 Test Cases Format 21

2.2.3 General Rules for eUICC Testing 27

2.2.4 VOID 31

2.2.5 Pass Criteria 31

2.2.6 Future Study 32

3 Testing Architecture 32

3.1 Testing Scope 32

3.2 Testing Execution 33

3.2.1 eUICC - Test Environment 33

3.2.2 VOID 34

3.2.3 VOID 34

3.2.4 VOID 34

3.2.5 Integrated eUICC - Test Environment 34

4 Interface Compliance Testing 35

4.1 General Overview 35

4.2 eUICC Interfaces 35

4.2.1 ATR and ISD-R Selection 35

4.2.2 ES6 (Operator -- eUICC): UpdateMetadata 41

4.2.3 ES8+ (SM-DP+ -- eUICC): InitialiseSecureChannel 61

4.2.4 ES8+ (SM-DP+ -- eUICC): ConfigureISDP 64

4.2.5 ES8+ (SM-DP+ -- eUICC): StoreMetadata 67

4.2.6 ES8+ (SM-DP+ -- eUICC): ReplaceSessionKeys 87

4.2.7 ES8+ (SM-DP+ -- eUICC): LoadProfileElements 89

4.2.8 ES10a (LPA -- eUICC): GetEuiccConfiguredData 100

4.2.9 ES10a (LPA -- eUICC): SetDefaultDpAddress 101

4.2.10 ES10b (LPA -- eUICC): PrepareDownload 103

4.2.11 ES10b (LPA -- eUICC): LoadBoundProfilePackage 108

4.2.12 ES10b (LPA -- eUICC): GetEUICCChallenge 114

4.2.13 ES10b (LPA -- eUICC): GetEUICCInfo 114

4.2.14 ES10b (LPA -- eUICC): ListNotification 122

4.2.15 ES10b (LPA -- eUICC): RetrieveNotificationsList 133

4.2.16 ES10b (LPA -- eUICC): RemoveNotificationFromList 153

4.2.17 ES10b (LPA -- eUICC): LoadCRL 159

4.2.18 ES10b (LPA -- eUICC): AuthenticateServer 159

4.2.19 ES10b (LPA -- eUICC): CancelSession 202

4.2.20 ES10c (LPA -- eUICC): GetProfilesInfo 210

4.2.21 ES10c (LPA -- eUICC): EnableProfile 223

4.2.22 ES10c (LPA -- eUICC): DisableProfile 346

4.2.23 ES10c (LPA -- eUICC): DeleteProfile 460

4.2.24 ES10c (LPA -- eUICC): eUICCMemoryReset 471

4.2.25 ES10c (LPA -- eUICC): GetEID 518

4.2.26 ES10c (LPA -- eUICC): SetNickname 519

4.2.27 ES10b (LPA -- eUICC): GetRAT 523

4.2.28 ES10b (LPA -- eUICC): LoadRPMPackage 524

4.3 VOID 602

4.4 VOID 602

4.5 VOID 602

4.6 VOID 602

4.7 LPAe Interfaces 602

5 Procedure - Behaviour Testing 602

5.1 General Overview 602

5.2 eUICC Behaviour 602

5.2.1 Retry mechanism 602

5.2.2 Forbidden PPRs 607

5.2.3 eUICC's RAT 609

5.2.4 eUICC File Structure 610

5.2.5 eUICC Delete Profile Process 611

5.2.6 eUICC Enable Profile Process 612

5.2.7 eUICC Disable Profile Process 613

5.2.8 eUICC Notifications 615

5.3 VOID 616

5.4 VOID 616

6 VOID 616

7 VOID 616

Annex A Constants 617

A.1 Generic Constants 617

A.2 Test Certificates and Test Keys 622

Annex B Dynamic Content 625

Annex C Methods And Procedures 631

C.1 Methods 631

C.2 Procedures 657

Annex D Commands And Responses 664

D.1 ES8+ Requests And Responses 664

D.1.1 ES8+ Requests 664

D.2 VOID 680

D.3 ES10x Requests And Responses 680

D.3.1 ES10x Requests 680

D.3.2 ES10x Responses 707

D.4 APDU 741

D.4.1 APDU Commands 741

D.4.2 R-APDU Chaining 741

D.5 ES6 Requests And Responses 742

D.5.1 ES6 Requests 742

D.6 VOID 747

D.7 VOID 747

D.8 VOID 747

D.9 VOID 747

Annex E Profiles 748

Annex F eUICC Settings 774

F.1 eUICC Settings 774

F.2 VOID 775

F.3 VOID 775

F.4 VOID 775

Annex G Initial States 776

G.1 VOID 776

G.2 eUICC 776

G.2.1 Common Initial States 776

G.2.2 For eUICC supporting NIST P-256 777

G.2.3 For eUICC supporting BrainpoolP256r1 777

G.2.4 With RAT configuration for SEP 777

G.2.5 With Additional PPARs in the RAT 777

G.2.6 Clean-up procedure 778

G.2.7 With RAT configuration for MEP 778

G.3 VOID 778

Annex H Icons and QR Codes 779

Annex J Integrated eUICC Testing (Normative) 781

J.1 Overview (Informative) 781

J.2 Integrated eUICC test requirements 782

J.3 USB CCID test interface 782

Annex L Document Management 784

L.1 Document History 784

L.2 Other Information 784

# Introduction

## 1.1 Overview

The main aim of the GSMA Remote SIM Provisioning specifications [2] & [3] is to provide solution for the Remote SIM Provisioning of Consumer Devices. The adoption of this technical solution will provide the basis for global interoperability between different Operator deployment scenarios, for example network equipment (e.g. Subscription Manager Data Preparation (SM-DP+)) and various eUICC platforms.

This Test Plan provides a set of test cases to be used for testing the eUICC implementations related to the provisioning system specifications documents [2] & [3]. This document offers an unified test strategy for checking the eUICC component.

## 1.2 Scope

This document is intended for:

* Parties which develop test tools
* Vendors (eUICC Manufacturers)
* Operators

The Test Plan consists of a set of relevant test cases for testing the eUICC. The only Implementation Under Test (IUT) within this document is the eUICC. The testing scopes developed in this document are:

* Interface compliance testing: Test cases to verify the compliance of the eUICC
* Interfaces System behaviour testing: Test cases to verify the functional behaviour of the eUICC

Each test case specified within this Test Plan refers to one or more requirements.

The Test Plan contains test cases for the following versions of SGP.22:

* GSMA RSP Technical Specification V3.1 [2]

This document includes an applicability table providing an indication whether test cases are relevant for a specific eUICC.

## 1.3 Definition of Terms

In addition to the terms which are defined below, the terms defined in SGP.22 [2] also apply.

| Term | Description |
| --- | --- |
| End User | The person using the Device. |
| Integrated eUICC Test Interface | An external interface provided by its manufacturer for the purpose of testing eUICC functionality. |
| Standalone Device | A Device which provides all the capabilities to be able to be used in an RSP environment and needs no other Device for the purpose of Remote SIM Provisioning. |
| Test Plan | Current document describing the test cases that allow the RSP ecosystem to be tested. |

## 1.4 Abbreviations

In addition to the abbreviations which are defined below, the abbreviations defined in SGP.22 [2] also apply.

| Abbreviation | Description |
| --- | --- |
| APDU | Application Protocol Data Unit |
| ATR | Answer To Reset |
| C-APDU | Command APDU |
| CCID | (USB) Chip Card Interface Device |
| CERT.CI.SIG | Certificate of the CI for its Public ECDSA Key |
| CERT.DPauth.SIG | Certificate of the SM-DP+ for its Public ECDSA key used for SM‑DP+ authentication |
| CERT.DPpb.SIG | Certificate of the SM-DP+ for its Public ECDSA key used for Profile Package Binding |
| CERT.DSauth.SIG | Certificate of the SM-DS for its Public ECDSA key used for SM‑DS authentication |
| CERT.EUICC.SIG | Certificate of the eUICC for its Public ECDSA key |
| CERT.EUM.SIG | Certificate of the EUM for its Public ECDSA key |
| DER TLV | Distinguished Encoding Rules - Tag Length Value |
| DPI | Delegated Platform Identifier |
| FCP | File Control Parameters |
| HW | Hardware |
| IUT | Implementation Under Test |
| KVN | Key Version Number |
| OCE | Off-Card Entity |
| OS | Operating System |
| PCMP | Profile Content Management Platform |
| PIR | Profile Installation Result |
| POR | Proof Of Receipt |
| R-APDU | Response APDU |
| RPR | Load RPM Package Result |
| SK.CI.SIG | Private key of the CI for signing certificates |
| SK.DPauth.SIG | Private Key of the of SM-DP+ for creating signatures for SM-DP+ authentication |
| SK.EUICC.SIG | Private key of the eUICC for creating signatures |
| SK.EUM.SIG | Private key of the EUM for creating signatures |
| SoC | System on a Chip |
| SP | Service Provider |
| SSD | Supplemental Security Domain |
| USB | Universal Serial Bus |

## 1.5 Document Cross-references

| Ref | Document Number | Title |
| --- | --- | --- |
| [1] | SGP.02 | GSMA "Remote Provisioning of Embedded UICC Technical specification" V3.1 |
| [2] | SGP.22 | RSP Technical Specification V3.1 |
| [3] | SGP.21 | RSP Architecture V3.1 |
| [4] | eUICC Profile Package | Trusted Connectivity Alliance (formerly SIMalliance) eUICC Profile Package: Interoperable Format Technical Specification V2.3.1 or later |
| [5] | ETSI TS 102 221 | Smart Cards; UICC-Terminal interface |
| [6] | GPC\_SPE\_034 | GlobalPlatform Card Specification v.2.3 |
| [7] | ISO/IEC 7816-4:2013 | Identification cards – Integrated circuit cards - Part 4: Organization, security and commands for interchange |
| [8] | RFC 5639 | Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation |
| [9] | ANSSI ECC FRP256V1 | Avis relatif aux paramètres de courbes elliptiques définis par l'Etat français. JORF n°0241 du 16 octobre 2011 page 17533. texte n° 30. 2011 |
| [10] | ITU E.118 | The international telecommunication charge card |
| [11] | NIST SP 800-56A | NIST Special Publication SP 800-56A: Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography (Revision 2), May 2013 |
| [12] | 3GPP TS 23.003 | Digital cellular telecommunications system (Phase 2+);  Universal Mobile Telecommunications System (UMTS);  Numbering, addressing and identification |
| [13] | ETSI TS 102 225 | Secured packet structure for UICC based applications; Release 12 |
| [14] | ETSI TS 102 226 | Remote APDU structure for UICC based applications; Release 9 |
| [15] | TS.26 | GSMA NFC Handset Requirements V9.0 |
| [16] | ITU-T X.690 (11/2008) | ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER) including Corrigendum 1 and 2 |
| [17] | ETSI TS 102 241 | Smart cards; UICC Application Programming Interface (UICC API) for Java Card™ |
| [18] | 3GPP TS 31.102 | Characteristics of the Universal Subscriber Identity Module (USIM) application |
| [19] | GPC\_SPE\_095 | GlobalPlatform Card - Digital Letter of Approval - Version 1.0 |
| [20] | RFC 2119 | Key words for use in RFCs to Indicate Requirement Levels, S. Bradner  <http://www.ietf.org/rfc/rfc2119.txt> |
| [22] | 3GPP TS 23.040 | Technical realization of the Short Message Service (SMS) |
| [23] | TCA Test | Trusted Connectivity Alliance (TCA) eUICC Profile Package: Interoperable Format Test Specification. See Annex K for applicable version. |
| [24] | RFC 4492 | Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) |
| [25] | SGP.26 | RSP Test Certificates Definition v3.0.X |
| [26] | Void | Void |
| [27] | Void | Void |
| [28] | CCID Rev 1.1 | CCID Specification for Integrated Circuit(s) Cards Interface Devices |
| [29] | SM2 algorithm | ISO/IEC 14888-3:2018 IT Security techniques – Digital signatures with appendix – Part 3: Discrete logarithm based mechanisms |

## 1.6 Conventions

The key words "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", and "MAY" in this document SHALL be interpreted as described in RFC 2119 [20].

# 2 Testing Rules

## 2.1 Applicability

### 2.1.1 Format of the Optional Features Table

The columns in Table 4 have the following meaning:

|  |  |
| --- | --- |
| Column | Meaning |
| Option | The optional feature supported or not by the implementation. |
| Mnemonic | The mnemonic column contains mnemonic identifiers for each item. |

Table 1: Format of the Optional Features Table

### 2.1.2 Format of the Applicability Table

The applicability of every test in Table 5 is formally expressed by the use of a Boolean expression defined in the following clause.

The columns in Table 5have the following meaning:

|  |  |
| --- | --- |
| Column | Meaning |
| Test case | The "Test case" column gives a reference to the test case number detailed in the present document and is required to validate the implementation of the corresponding item in the "Name" column. |
| Name | In the "Name" column, a short non-exhaustive description of the test is found. |
| Version | This column indicates which test cases are applicable for the given SGP.22 version.  See clause 2.1.3 'Applicability and Notations'. |
| Test Env. | Test environment used for executing the test case. |

Table 2: Format of the Applicability Table

### 2.1.3 Applicability and Notations

The following notations are used for the Applicability column:

| Applicability code | Meaning |
| --- | --- |
| M | mandatory - the capability is required to be supported. |
| N/A | not applicable - in the given context, it is impossible to use the capability. |
| Ci | conditional - the requirement on the capability depends on the support of other items. "i" is an integer identifying an unique conditional status expression which is defined immediately following the table. For nested conditional expressions, the syntax "IF ... THEN (IF ... THEN ... ELSE...) ELSE ..." is to be used to avoid ambiguities. |

Table 3: Applicability and Notations

### 2.1.4 Optional Features Table

The supplier of the implementation SHALL state the support of possible options in Table 5.

|  |  |
| --- | --- |
| eUICC Options | Mnemonic |
| The eUICC supports NIST P-256 [11] for signing and for verification (see NOTE 1) | O\_E\_NIST |
| The eUICC supports brainpoolP256r1 [8] for signing and for verification (see NOTE 1) | O\_E\_BRP |
| The eUICC supports FRP256V1 [9] for signing and for verification (see NOTE 1) | O\_E\_FRP |
| The eUICC supports SM2 [XX] for signing and for verification | O\_E\_SM2 |
| The eUICC supports the LPAe | O\_E\_LPAe |
| The eUICC supports LPA Proxy | O\_E\_LPA\_PROXY |
| The eUICC stores the otPK.eUICC.ECKA / otSK.eUICC.ECKA from previous unsuccessful download attempt for future retry | O\_E\_REUSE\_OTPK |
| The eUICC can hold two PIR | O\_E\_2\_PIR |
| The eUICC supports RPM | O\_E\_RPM |
| The eUICC supports Enterprise Profiles | O\_E\_ENTERPRISE |
| The eUICC terminates EnableProfile and DisableProfile with error "catBusy" when a proactive session is ongoing and the refresh flag is set | O\_E\_CATBUSY\_EN\_DIS\_REFRESH |
| The eUICC terminates EnableProfile and DisableProfile with error "catBusy" when a proactive session is ongoing and the refresh Flag is not set | O\_E\_CATBUSY\_EN\_DIS\_NOREFRESH |
| The eUICC terminates eUICCMemoryReset with error "catBusy" when a proactive session is ongoing | O\_E\_CATBUSY\_MR |
| The eUICC is based on an integrated TRE | O\_E\_INTEGRATED |
| The eUICC supports variant O for signing | O\_VAR\_O |
| The eUICC supports variant Ov3 for signing | O\_VAR\_OV3 |
| The eUICC supports variant A for signing | O\_VAR\_A |
| The eUICC supports variant B for signing | O\_VAR\_B |
| The eUICC supports variant C for signing | O\_VAR\_C |
| The eUICC supports the HRI server address in the Profile Metadata | O\_E\_HRI\_ADDRESS\_IN\_PM |
| The eUICC supports the service description in the Profile Metadata | O\_E\_SERVICE\_DESCRIPTION\_IN\_PM |
| The eUICC supports the features defined for Device Change and Profile Recovery | O\_E\_DEVICE\_CHANGE |
| The eUICC accepts an estimated Profile size in the Profile Metadata | O\_E\_PROFILE\_SIZE\_IN\_PM |
| The eUICC provides an estimated Profile size in ES10c.GetProfilesInfo | O\_E\_PROFILE\_SIZE\_IN\_PROFILE\_INFO |
| The eUICC supports the OS Update capability | O\_E\_OS\_UPDATE |
| The eUICC supports “catBusy” error code | O\_E\_CATBUSY |
| The eUICC supports MEP | O\_E\_MEP |
| The eUICC supports MEP-A1 | O\_E\_MEP\_A1 |
| The eUICC supports MEP-A2 | O\_E\_MEP\_A2 |
| The eUICC supports MEP-B | O\_E\_MEP\_B |
| The eUICC supports Refresh Flag not set for MEP-B | O\_E\_MEP\_B\_NO\_REFRESH |
| The eUICC supports profiles with non-IMSI SUPI Type | O\_E\_NON\_IMSI\_SUPI\_TYPE |
| The eUICC supports eUICC Profile Package Specification v3.x | O\_E\_ADD\_PP\_VERSIONS |
| The eUICC supports the extensibility in the DeviceInfo | O\_E\_DEVICE\_INFO\_EXTENSIBILITY\_SUPPORT |
| * NOTE 1: This version of test specification extensively tests an eUICC that supports O\_E\_NIST or O\_E\_BRP. Any O\_E\_FRP or O\_E\_SM2 test cases are for further study. | |

Table 4: Options

### 2.1.5 Applicability Table

Table 5 specifies the applicability of each test case. See clause 2.1.2 for the format of this table.

| Test case | Name | V3.1 | Test Env. |
| --- | --- | --- | --- |
| eUICC Interfaces Compliance Testing | | | |
| 4.2.1.2.1 | TC\_eUICC\_ATR\_And\_ISDR\_Selection  Only the test sequence #1 | C006 | TE\_eUICC |
| 4.2.1.2.1 | TC\_eUICC\_ATR\_And\_ISDR\_Selection  Only the test sequence #2 | M | TE\_eUICC |
| 4.2.1.2.1 | TC\_eUICC\_ATR\_And\_ISDR\_Selection  Only the test sequence #3 | C325 | TE\_eUICC |
| 4.2.1.2.1 | TC\_eUICC\_ATR\_And\_ISDR\_Selection  Only the test sequences #4 and #7 | C318 | TE\_eUICC |
| 4.2.1.2.1 | TC\_eUICC\_ATR\_And\_ISDR\_Selection  Only the test sequences #5 and #8 | C328 | TE\_eUICC |
| 4.2.1.2.1 | TC\_eUICC\_ATR\_And\_ISDR\_Selection  Only the test sequences #6 and #9 | C327 | TE\_eUICC |
| 4.2.2.2.1 | TC\_eUICC\_ES6.UpdateMetadata | C319 | TE\_eUICC |
| 4.2.2.2.2 | TC\_eUICC\_ES6.UpdateMetadata\_EnterpriseProfiles | C309 | TE\_eUICC |
| 4.2.2.2.3 | TC\_eUICC\_ES6.UpdateMetadata\_Service\_Specific\_Data | C319 | TE\_eUICC |
| 4.2.2.2.4 | TC\_eUICC\_ES6.UpdateMetadata\_V3NotificationConfiguration | C319 | TE\_eUICC |
| 4.2.2.2.5 | TC\_eUICC\_ES6.UpdateMetadata\_V3RPM | C305 | TE\_eUICC |
| 4.2.2.2.6 | TC\_eUICC\_ES6.UpdateMetadata\_V3HRIServerAddress | C355 | TE\_eUICC |
| 4.2.2.2.7 | TC\_eUICC\_ES6.UpdateMetadata\_V3LPRConfiguration | C354 | TE\_eUICC |
| 4.2.2.2.8 | TC\_eUICC\_ES6.UpdateMetadata\_V3DeviceChange | C352 | TE\_eUICC |
| 4.2.3.2.1 | TC\_eUICC\_ES8+.InitialiseSecureChannel | M | TE\_eUICC |
| 4.2.4.2.1 | TC\_eUICC\_ES8+.ConfigureISDP | M | TE\_eUICC |
| 4.2.5.2.1 | TC\_eUICC\_ES8+.StoreMetadata  All test sequences except the sequences #1, #5 and #11 | M | TE\_eUICC |
| 4.2.5.2.1 | TC\_eUICC\_ES8+.StoreMetadata  Only the test sequences #1 and #5 | C319 | TE\_eUICC |
| 4.2.5.2.1 | TC\_eUICC\_ES8+.StoreMetadata  Only the test sequence #11 | C356 | TE\_eUICC |
| 4.2.5.2.2 | TC\_eUICC\_ES8+.StoreMetadata\_Service\_Specific\_Data | M | TE\_eUICC |
| 4.2.5.2.3 | TC\_eUICC\_ES8+.StoreMetadata\_EnterpriseProfiles  All test sequences except the sequence #4 | C309 | TE\_eUICC |
| 4.2.5.2.3 | TC\_eUICC\_ES8+.StoreMetadata\_EnterpriseProfiles  Only the test sequence #4 | C349 | TE\_eUICC |
| 4.2.6.2.1 | TC\_eUICC\_ES8+.ReplaceSessionKeys | M | TE\_eUICC |
| 4.2.7.2.1 | TC\_eUICC\_ES8+.LoadProfileElements except test sequence #11 and #12 | M | TE\_eUICC |
| 4.2.7.2.1 | TC\_eUICC\_ES8+.LoadProfileElements  Only the test sequences #11 and #12 | C346 | TE\_eUICC |
| 4.2.8.2.1 | TC\_eUICC\_ES10a.GetEuiccConfiguredAddresses | M | TE\_eUICC |
| 4.2.9.2.1 | TC\_eUICC\_ES10a.SetDefaultDpAddress | M | TE\_eUICC |
| 4.2.10.2.1 | TC\_eUICC\_ES10b.PrepareDownloadNIST | C001 | TE\_eUICC |
| 4.2.10.2.2 | TC\_eUICC\_ES10b.PrepareDownloadBRP | C002 | TE\_eUICC |
| 4.2.10.2.3 | TC\_eUICC\_ES10b.PrepareDownloadFRP | C003 | TE\_eUICC |
| 4.2.10.2.4 | TC\_eUICC\_ES10b.PrepareDownloadErrorCases | M | TE\_eUICC |
| 4.2.11.2.1 | TC\_eUICC\_ES10b.LoadBoundProfilePackageNIST | C001 | TE\_eUICC |
| 4.2.11.2.2 | TC\_eUICC\_ES10b.LoadBoundProfilePackageBRP | C002 | TE\_eUICC |
| 4.2.11.2.3 | TC\_eUICC\_ES10b.LoadBoundProfilePackageFRP | C003 | TE\_eUICC |
| 4.2.11.2.4 | TC\_eUICC\_ES10b.LoadBoundProfilePackage\_ErrorCases | M | TE\_eUICC |
| 4.2.12.2.1 | TC\_eUICC\_ES10b.GetEUICCChallenge | M | TE\_eUICC |
| 4.2.13.2.1 | TC\_eUICC\_ES10b.GetEUICCInfo1  Only test sequences #01 and #02 | M | TE\_eUICC |
| 4.2.13.2.1 | TC\_eUICC\_ES10b.GetEUICCInfo1  Only test sequence #03 | C310 | TE\_eUICC |
| 4.2.13.2.1 | TC\_eUICC\_ES10b.GetEUICCInfo1  Only test sequence #04 | C317 | TE\_eUICC |
| 4.2.13.2.1 | TC\_eUICC\_ES10b.GetEUICCInfo1  Only test sequence #05 | C313 | TE\_eUICC |
| 4.2.13.2.1 | TC\_eUICC\_ES10b.GetEUICCInfo1  Only test sequence #06 | C314 | TE\_eUICC |
| 4.2.13.2.2 | TC\_eUICC\_ES10b.GetEUICCInfo2\_RSP\_V2.1 | N/A | TE\_eUICC |
| 4.2.13.2.3 | TC\_eUICC\_ES10b.GetEUICCInfo2\_RSP\_V2.2.x | M | TE\_eUICC |
| 4.2.13.2.4 | TC\_eUICC\_ES10b.GetEUICCInfo2 | C310 | TE\_eUICC |
| 4.2.13.2.8 | TC\_eUICC\_ES10b.GetEUICCInfo2\_RSP\_V3.X | M | TE\_eUICC |
| 4.2.13.2.9 | TC\_eUICC\_ES10b.GetEUICCInfo2\_RSP\_V3.x\_Integrated\_eUICC | C040 | TE\_eUICC |
| 4.2.14.2.1 | TC\_eUICC\_ES10b.ListNotification  All test sequences except the sequence #5 | M | TE\_eUICC |
| 4.2.14.2.1 | TC\_eUICC\_ES10b.ListNotification  Only the test sequence #5 | C025 | TE\_eUICC |
| 4.2.14.2.2 | TC\_eUICC\_ES10b.ListNotification\_RPM | C301 | TE\_eUICC |
| 4.2.15.2.1 | TC\_eUICC\_ES10b.RetrieveNotificationsList  All test sequences except the sequences #5 and #15 | M | TE\_eUICC |
| 4.2.15.2.1 | TC\_eUICC\_ES10b.RetrieveNotificationsList  Only the test sequences #5 and #15 | C025 | TE\_eUICC |
| 4.2.16.2.1 | TC\_eUICC\_ES10b.RemoveNotificationFromList  All test sequences except the sequence #5 | M | TE\_eUICC |
| 4.2.18.2.1 | TC\_eUICC\_ES10b.AuthenticateServer\_SM-DP+\_NIST\_Server\_Variant\_O | C357 | TE\_eUICC |
| 4.2.18.2.2 | TC\_eUICC\_ES10b.AuthenticateServer\_SM-DP+\_BRP\_Server\_Variant\_O | C358 | TE\_eUICC |
| 4.2.18.2.4 | TC\_eUICC\_ES10b.AuthenticateServer\_SM-DP+\_ErrorCases\_Server\_Variant\_O | M | TE\_eUICC |
| 4.2.18.2.5 | TC\_eUICC\_ES10b.AuthenticateServer\_SM-DS\_BRP\_Server\_Variant\_O | C358 | TE\_eUICC |
| 4.2.18.2.6 | TC\_eUICC\_ES10b.AuthenticateServer\_SM-DS\_NIST\_Server\_Variant\_O | C357 | TE\_eUICC |
| 4.2.18.2.8 | TC\_eUICC\_ES10b.AuthenticateServer\_SM-DS\_ErrorCases | M | TE\_eUICC |
| 4.2.18.2.9 | TC\_eUICC\_ES10b.AuthenticateServerV3\_SM-DP+\_NIST\_Server\_Variants\_V3 | C316 | TE\_eUICC |
| 4.2.18.2.10 | TC\_eUICC\_ES10b.AuthenticateServerV3\_SM-DP+\_ErrorCases\_V3\_Server\_Variant\_A | C315 | TE\_eUICC |
| 4.2.18.2.11 | TC\_eUICC\_ES10b.AuthenticateServer\_SM-DP+\_ErrorCases\_V3\_Server\_Variant\_B | C313 | TE\_eUICC |
| 4.2.18.2.12 | TC\_eUICC\_ES10b.AuthenticateServer\_SM-DP+\_ErrorCases\_V3\_Server\_Variant\_C | C314 | TE\_eUICC |
| 4.2.18.2.13 | TC\_eUICC\_ES10b.AuthenticateServerV3\_SM-DP+\_Independently\_Of\_Server\_Variant | M | TE\_eUICC |
| 4.2.18.2.14 | TC\_eUICC\_ES10b.AuthenticateServerV3\_SM-DS\_Independently\_Of\_Server\_Variant | M | TE\_eUICC |
| 4.2.19.2.1 | TC\_eUICC\_ES10b.CancelSessionNIST | C001 | TE\_eUICC |
| 4.2.19.2.2 | TC\_eUICC\_ES10b.CancelSessionBRP | C002 | TE\_eUICC |
| 4.2.19.2.3 | TC\_eUICC\_ES10b.CancelSessionFRP | C003 | TE\_eUICC |
| 4.2.19.2.4 | TC\_eUICC\_ES10b.CancelSession\_ErrorCase | M | TE\_eUICC |
| 4.2.20.2.1 | TC\_eUICC\_ES10c.GetProfilesInfo | M | TE\_eUICC |
| 4.2.20.2.2 | TC\_eUICC\_ES10c.GetProfilesInfo\_ErrorCases | M | TE\_eUICC |
| 4.2.20.2.3 | TC\_eUICC\_ES10c.GetProfilesInfo\_MEPA1 | C318 | TE\_eUICC |
| 4.2.20.2.4 | TC\_eUICC\_ES10c.GetProfilesInfo\_MEPA2 | C328 | TE\_eUICC |
| 4.2.20.2.5 | TC\_eUICC\_ES10c.GetProfilesInfo\_MEPB | C327 | TE\_eUICC |
| 4.2.21.2.1 | TC\_eUICC\_ES10c.EnableProfile\_Case3  All test sequences except the sequences #7, #8, #9 and #10 | C319 | TE\_eUICC |
| 4.2.21.2.1 | TC\_eUICC\_ES10c.EnableProfile\_Case3  Only the test sequences #7 and #9 | C033 | TE\_eUICC |
| 4.2.21.2.1 | TC\_eUICC\_ES10c.EnableProfile\_Case3  Only the test sequences #8 and #10 | C037 | TE\_eUICC |
| 4.2.21.2.2 | TC\_eUICC\_ES10c.EnableProfile\_ErrorCases\_Case3  All test sequences except the sequences #7 and #8 | C319 | TE\_eUICC |
| 4.2.21.2.2 | TC\_eUICC\_ES10c.EnableProfile\_ErrorCases\_Case3  Only the test sequence #7 | C036 | TE\_eUICC |
| 4.2.21.2.2 | TC\_eUICC\_ES10c.EnableProfile\_ErrorCases\_Case3  Only the test sequence #8 | C032 | TE\_eUICC |
| 4.2.21.2.3 | TC\_eUICC\_ES10c.EnableProfile\_Case4  All test sequences except the sequences #7, #8, #9 and #10 | C319 | TE\_eUICC |
| 4.2.21.2.3 | TC\_eUICC\_ES10c.EnableProfile\_Case4  Only the test sequences #7 and #9 | C033 | TE\_eUICC |
| 4.2.21.2.3 | TC\_eUICC\_ES10c.EnableProfile\_Case4  Only the test sequence #8 and #10 | C037 | TE\_eUICC |
| 4.2.21.2.4 | TC\_eUICC\_ES10c.EnableProfile\_ErrorCases\_Case4  All test sequences except the sequences #5, #6, #7 and #8 | C319 | TE\_eUICC |
| 4.2.21.2.4 | TC\_eUICC\_ES10c.EnableProfile\_ErrorCases\_Case4  Only the test sequences #5 and #6 | C319 | TE\_eUICC |
| 4.2.21.2.4 | TC\_eUICC\_ES10c.EnableProfile\_ErrorCases\_Case4  Only the test sequence #7 | C036 | TE\_eUICC |
| 4.2.21.2.4 | TC\_eUICC\_ES10c.EnableProfile\_ErrorCases\_Case4  Only the test sequence #8 | C032 | TE\_eUICC |
| 4.2.21.2.5 | TC\_eUICC\_ES10c.EnableProfile\_ErrorCases\_Case4\_catBusySupported | C032 | TE\_eUICC |
| 4.2.21.2.6 | TC\_eUICC\_ES10c.EnableProfile\_Case4\_catBusyNotSupported | C033 | TE\_eUICC |
| 4.2.21.2.7 | TC\_eUICC\_ES10c.EnableProfile\_ErrorCases\_Case3\_catBusySupported | C032 | TE\_eUICC |
| 4.2.21.2.8 | TC\_eUICC\_ES10c.EnableProfile\_Case4\_MEPA1  All test sequences except the sequences #7, #8, #15, #16, #17 and #18 | C318 | TE\_eUICC |
| 4.2.21.2.8 | TC\_eUICC\_ES10c.EnableProfile\_Case4\_MEPA1  Only the test sequences #7, #15, #16, #17 and #18 | C333 | TE\_eUICC |
| 4.2.21.2.8 | TC\_eUICC\_ES10c.EnableProfile\_Case4\_MEPA1  Only the test sequence #8 | C329 | TE\_eUICC |
| 4.2.21.2.9 | TC\_eUICC\_ES10c.EnableProfile\_ErrorCases\_Case4\_MEPA1  All test sequences except the sequences #5, #6 and #11 | C318 | TE\_eUICC |
| 4.2.21.2.9 | TC\_eUICC\_ES10c.EnableProfile\_ErrorCases\_Case4\_MEPA1  Only the test sequences #6 and #11 | C332 | TE\_eUICC |
| 4.2.21.2.9 | TC\_eUICC\_ES10c.EnableProfile\_ErrorCases\_Case4\_MEPA1  Only the test sequence #5 | C336 | TE\_eUICC |
| 4.2.21.2.10 | TC\_eUICC\_ES10c.EnableProfile\_Case4\_MEPA2 | C328 | TE\_eUICC |
| 4.2.21.2.11 | TC\_eUICC\_ES10c.EnableProfile\_ErrorCases\_Case4\_MEPA2  All test sequences except the sequences #5 and #10 | C328 | TE\_eUICC |
| 4.2.21.2.11 | TC\_eUICC\_ES10c.EnableProfile\_ErrorCases\_Case4\_MEPA2  Only the test sequence #5 and #10 | C330 | TE\_eUICC |
| 4.2.21.2.12 | TC\_eUICC\_ES10c.EnableProfile\_Case4\_MEPB  All test sequences except the sequences #5, #6, #7, #8, #13, #14, #15 and #16 | C327 | TE\_eUICC |
| 4.2.21.2.12 | TC\_eUICC\_ES10c.EnableProfile\_Case4\_MEP\_B  Only the test sequences #5, #6, #13, #14 | C345 | TE\_eUICC |
| 4.2.21.2.12 | TC\_eUICC\_ES10c.EnableProfile\_Case4\_MEPB  Only the test sequences #7, #15 and #16 | C335 | TE\_eUICC |
| 4.2.21.2.12 | TC\_eUICC\_ES10c.EnableProfile\_Case4\_MEPB  Only the test sequence #8 | C337 | TE\_eUICC |
| 4.2.21.2.13 | TC\_eUICC\_ES10c.EnableProfile\_ErrorCases\_Case4\_MEPB  All test sequences except the sequences #5, #6 and #11 | C327 | TE\_eUICC |
| 4.2.21.2.13 | TC\_eUICC\_ES10c.EnableProfile\_ErrorCases\_Case4\_MEPB  Only the test sequences #6 and #11 | C334 | TE\_eUICC |
| 4.2.21.2.13 | TC\_eUICC\_ES10c.EnableProfile\_ErrorCases\_Case4\_MEPB  Only the test sequence #5 | C338 | TE\_eUICC |
| 4.2.22.2.1 | TC\_eUICC\_ES10c.DisableProfile\_Case3  All test sequences except the sequences #7, sequence #8, #9 and #10 | C319 | TE\_eUICC |
| 4.2.22.2.1 | TC\_eUICC\_ES10c.DisableProfile\_Case3  Only the test sequences #7 and #9 | C033 | TE\_eUICC |
| 4.2.22.2.1 | TC\_eUICC\_ES10c.DisableProfile\_Case3  Only the test sequences #8 and #10 | C037 | TE\_eUICC |
| 4.2.22.2.2 | TC\_eUICC\_ES10c.DisableProfile\_ErrorCases\_Case3  All test sequences except the sequences #7 and #8 | C319 | TE\_eUICC |
| 4.2.22.2.2 | TC\_eUICC\_ES10c.DisableProfile\_ErrorCases\_Case3  Only the test sequence #7 | C036 | TE\_eUICC |
| 4.2.22.2.2 | TC\_eUICC\_ES10c.DisableProfile\_ErrorCases\_Case3  Only the test sequence #8 | C032 | TE\_eUICC |
| 4.2.22.2.3 | TC\_eUICC\_ES10c.DisableProfile\_Case4  All test sequences except the sequences #7, #8, #9 and #10 | C319 | TE\_eUICC |
| 4.2.22.2.3 | TC\_eUICC\_ES10c.DisableProfile\_Case4  Only the test sequences #7 and #9 | C033 | TE\_eUICC |
| 4.2.22.2.3 | TC\_eUICC\_ES10c.DisableProfile\_Case4  Only the test sequences #8 and #10 | C037 | TE\_eUICC |
| 4.2.22.2.4 | TC\_eUICC\_ES10c.DisableProfile\_ErrorCases\_Case4  All test sequences except the sequences #5, #6, #7 and #8 | C319 | TE\_eUICC |
| 4.2.22.2.4 | TC\_eUICC\_ES10c.DisableProfile\_ErrorCases\_Case4  Only the test sequences #5 and #6 | C319 | TE\_eUICC |
| 4.2.22.2.4 | TC\_eUICC\_ES10c.DisableProfile\_ErrorCases\_Case4  Only the test sequence #7 | C036 | TE\_eUICC |
| 4.2.22.2.4 | TC\_eUICC\_ES10c.DisableProfile\_ErrorCases\_Case4  Only the test sequence #8 | C032 | TE\_eUICC |
| 4.2.22.2.5 | TC\_eUICC\_ES10c.DisableProfile\_Case4\_MEPA1  All test sequences except the sequences #7, #8, #15, #16, #17 and #18 | C318 | TE\_eUICC |
| 4.2.22.2.5 | TC\_eUICC\_ES10c.DisableProfile\_Case4\_MEPA1  Only the test sequences #7, #15, #16, #17 and #18 | C333 | TE\_eUICC |
| 4.2.22.2.5 | TC\_eUICC\_ES10c.DisableProfile\_Case4\_MEPA1  Only the test sequence #8 | C329 | TE\_eUICC |
| 4.2.22.2.6 | TC\_eUICC\_ES10c.DisableProfile\_ErrorCases\_Case4\_MEPA1  All test sequences except the sequences #5, #6 and #11 | C318 | TE\_eUICC |
| 4.2.22.2.6 | TC\_eUICC\_ES10c.DisableProfile\_ErrorCases\_Case4\_MEPA1  Only the test sequences #6 and #11 | C332 | TE\_eUICC |
| 4.2.22.2.6 | TC\_eUICC\_ES10c.DisableProfile\_ErrorCases\_Case4\_MEPA1  Only the test sequence #5 | C336 | TE\_eUICC |
| 4.2.22.2.7 | TC\_eUICC\_ES10c.DisableProfile\_Case4\_MEPA2  All test sequences except the sequence #3 | C328 | TE\_eUICC |
| 4.2.22.2.7 | TC\_eUICC\_ES10c.DisableProfile\_Case4\_MEPA2  Only the test sequence #3 | C331 | TE\_eUICC |
| 4.2.22.2.8 | TC\_eUICC\_ES10c.DisableProfile\_ErrorCases\_Case4\_MEPA2  All test sequences except the sequences #5 and #10 | C328 | TE\_eUICC |
| 4.2.22.2.8 | TC\_eUICC\_ES10c.DisableProfile\_ErrorCases\_Case4\_MEPA2  Only the test sequences #5 and #10 | C330 | TE\_eUICC |
| 4.2.22.2.9 | TC\_eUICC\_ES10c.DisableProfile\_Case4\_MEPB  All test sequences except the sequences #5, #6, #7, #8, #13, #14, #15 #16, #17 and #18 | C327 | TE\_eUICC |
| 4.2.22.2.9 | TC\_eUICC\_ES10c.DisableProfile\_Case4\_MEPB  All test sequences except the sequences #5, #6, #13, #14 | C345 | TE\_eUICC |
| 4.2.22.2.9 | TC\_eUICC\_ES10c.DisableProfile\_Case4\_MEPB  Only the test sequences #7, #15, #16, #17 and #18 | C335 | TE\_eUICC |
| 4.2.22.2.9 | TC\_eUICC\_ES10c.DisableProfile\_Case4\_MEPB  Only the test sequence #8 | C337 | TE\_eUICC |
| 4.2.22.2.10 | TC\_eUICC\_ES10c.DisableProfile\_ErrorCases\_Case4\_MEPB  All test sequences except the sequences #5, #6 and #11 | C327 | TE\_eUICC |
| 4.2.22.2.10 | TC\_eUICC\_ES10c.DisableProfile\_ErrorCases\_Case4\_MEPB  Only the test sequences #6 and #11 | C334 | TE\_eUICC |
| 4.2.22.2.10 | TC\_eUICC\_ES10c.DisableProfile\_ErrorCases\_Case4\_MEPB  Only the test sequence #5 | C338 | TE\_eUICC |
| 4.2.23.2.1 | TC\_eUICC\_ES10c.DeleteProfile\_Case3 | M | TE\_eUICC |
| 4.2.23.2.2 | TC\_eUICC\_ES10c.DeleteProfile\_ErrorCases\_Case3 | M | TE\_eUICC |
| 4.2.23.2.3 | TC\_eUICC\_ES10c.DeleteProfile\_Case4 | M | TE\_eUICC |
| 4.2.23.2.4 | TC\_eUICC\_ES10c.DeleteProfile\_ErrorCases\_Case4 | M | TE\_eUICC |
| 4.2.24.2.1 | TC\_eUICC\_ES10c.eUICCMemoryReset  All test sequences except the sequences #2, #5 and #6 | C319 | TE\_eUICC |
| 4.2.24.2.1 | TC\_eUICC\_ES10c.eUICCMemoryReset  Only the test sequence #2 | C319 | TE\_eUICC |
| 4.2.24.2.1 | TC\_eUICC\_ES10c.eUICCMemoryReset  Only the test sequences #5 and #6 | C039 | TE\_eUICC |
| 4.2.24.2.2 | TC\_eUICC\_ES10c.eUICCMemoryReset\_ErrorCases  Only the test sequence #1 | C038 | TE\_eUICC |
| 4.2.24.2.2 | TC\_eUICC\_ES10c.eUICCMemoryReset\_ErrorCases  Only the test sequence #2 | C319 | TE\_eUICC |
| 4.2.24.2.3 | TC\_eUICC\_ES10c.eUICCMemoryReset\_MEPA1  All test sequences except the sequences #6, #7, #8 and #9 | C318 | TE\_eUICC |
| 4.2.24.2.3 | TC\_eUICC\_ES10c.eUICCMemoryReset\_MEPA1  Only the test sequences #6, #7, #8 and #9 | C326 | TE\_eUICC |
| 4.2.24.2.4 | TC\_eUICC\_ES10c.eUICCMemoryReset\_ErrorCases\_MEPA1  All test sequences except the sequence #2 | C318 | TE\_eUICC |
| 4.2.24.2.4 | TC\_eUICC\_ES10c.eUICCMemoryReset\_ErrorCases\_MEPA1  Only the test sequence #2 | C324 | TE\_eUICC |
| 4.2.24.2.5 | TC\_eUICC\_ES10c.eUICCMemoryReset\_MEPA2  All test sequences except the sequences #6, #7, #8 and #9 | C328 | TE\_eUICC |
| 4.2.24.2.5 | TC\_eUICC\_ES10c.eUICCMemoryReset\_MEPA2  Only the test sequences #6, #7, #8 and #9 | C340 | TE\_eUICC |
| 4.2.24.2.6 | TC\_eUICC\_ES10c.eUICCMemoryReset\_ErrorCases\_MEPA2  All test sequences except the sequence #2 | C328 | TE\_eUICC |
| 4.2.24.2.6 | TC\_eUICC\_ES10c.eUICCMemoryReset\_ErrorCases\_MEPA2  Only the test sequence #2 | C339 | TE\_eUICC |
| 4.2.24.2.7 | TC\_eUICC\_ES10c.eUICCMemoryReset\_MEPB  All test sequences except the sequences #6, #7, #8 and #9 | C327 | TE\_eUICC |
| 4.2.24.2.7 | TC\_eUICC\_ES10c.eUICCMemoryReset\_MEPB  Only the test sequences #6, #7, #8 and #9 | C342 | TE\_eUICC |
| 4.2.24.2.8 | TC\_eUICC\_ES10c.eUICCMemoryReset\_ErrorCases\_MEPB  All test sequences except the sequence #2 | C327 | TE\_eUICC |
| 4.2.24.2.8 | TC\_eUICC\_ES10c.eUICCMemoryReset\_ErrorCases\_MEPB  Only the test sequence #2 | C341 | TE\_eUICC |
| 4.2.25.2.1 | TC\_eUICC\_ES10c.GetEID | M | TE\_eUICC |
| 4.2.26.2.1 | TC\_eUICC\_ES10c.SetNickname | M | TE\_eUICC |
| 4.2.27.2.1 | TC\_eUICC\_ES10b.GetRAT  Only the test Sequence #1 | C319 | TE\_eUICC |
| 4.2.27.2.1 | TC\_eUICC\_ES10b.GetRAT  Only the test Sequence #2 | C319 | TE\_eUICC |
| 4.2.28.2.1 | TC\_eUICC\_ES10b.LoadRPMPackage\_EnableProfile  All test sequences except the sequences #5 and #7 | C301 | TE\_eUICC |
| 4.2.28.2.1 | TC\_eUICC\_ES10b.LoadRPMPackage\_EnableProfile  Only the test sequences #5 and #7 | C305 | TE\_eUICC |
| 4.2.28.2.2 | TC\_eUICC\_ES10b.LoadRPMPackage\_DisableProfile  All test sequences except the sequence #6 | C301 | TE\_eUICC |
| 4.2.28.2.2 | TC\_eUICC\_ES10b.LoadRPMPackage\_DisableProfile  Only the test sequence #6 | C305 | TE\_eUICC |
| 4.2.28.2.3 | TC\_eUICC\_ES10b.LoadRPMPackage\_DeleteProfile | C301 | TE\_eUICC |
| 4.2.28.2.4 | TC\_eUICC\_ES10b.LoadRPMPackage\_ListProfileInfo | C305 | TE\_eUICC |
| 4.2.28.2.5 | TC\_eUICC\_ES10b.LoadRPMPackage\_UpdateMetadata  All test sequences except the sequence #9 | C305 | TE\_eUICC |
| 4.2.28.2.5 | TC\_eUICC\_ES10b.LoadRPMPackage\_UpdateMetadata  Only the test sequence #9 | C301 | TE\_eUICC |
| 4.2.28.2.6 | TC\_eUICC\_ES10b.LoadRPMPackage\_ContactPCMP  All test sequences except the sequence #4 | C303 | TE\_eUICC |
| 4.2.28.2.6 | TC\_eUICC\_ES10b.LoadRPMPackage\_ContactPCMP  Only the test sequence #4 | C350 | TE\_eUICC |
| 4.2.28.2.7 | TC\_eUICC\_ES10b.LoadRPMPackage\_Multiple\_RPM  All test sequences except the sequences #3, #4, #5, and #7 | C301 | TE\_eUICC |
| 4.2.28.2.7 | TC\_eUICC\_ES10b.LoadRPMPackage\_Multiple\_RPM  Only the test sequences #3, #4 and #5 | C350 | TE\_eUICC |
| 4.2.28.2.7 | TC\_eUICC\_ES10b.LoadRPMPackage\_Multiple\_RPM  Only the test sequence #7 | C303 | TE\_eUICC |
| 4.2.28.2.8 | TC\_eUICC\_ES10b.LoadRPMPackage\_ErrorCases | C301 | TE\_eUICC |
| 4.2.28.2.9 | TC\_eUICC\_ES10b.LoadRPMPackage\_Enterprise\_Profiles  All test sequences except the sequence #13 | C302 | TE\_eUICC |
| 4.2.28.2.9 | TC\_eUICC\_ES10b.LoadRPMPackage\_Enterprise\_Profiles  Only the test sequence #13 | C351 | TE\_eUICC |
| 4.2.28.2.10 | TC\_eUICC\_ES10b.LoadRPMPackage\_CatBusy | C304 | TE\_eUICC |
| Procedure - Behaviour Testing | | | |
| 5.2.1.2.1 | TC\_eUICC\_PrepareDownload\_Retry\_ReuseOTKeys | C019 | TE\_eUICC |
| 5.2.1.2.2 | TC\_eUICC\_PrepareDownload\_Retry\_NewOTKeys | C020 | TE\_eUICC |
| 5.2.2.2.1 | TC\_eUICC\_ForbiddenPPRs | C319 | TE\_eUICC |
| 5.2.3.2.1 | TC\_eUICC\_GetProfilesInfo\_GetRAT\_RSPSession | C319 | TE\_eUICC |
| 5.2.4.2.1 | TC\_eUICC\_Default\_FileSystem | M | TE\_eUICC |
| 5.2.5.2.1 | TC\_eUICC\_DeleteProfile\_ISDP\_And\_Components | M | TE\_eUICC |
| 5.2.6.2.1 | TC\_eUICC\_EnableProfile\_Twice\_Notifications | M | TE\_eUICC |
| 5.2.7.2.1 | TC\_eUICC\_DisableProfile\_ApplicationManagement | M | TE\_eUICC |
| 5.2.8.2.1 | TC\_eUICC\_Enable\_Disable\_Delete\_Notifications | M | TE\_eUICC |
| Test Specifications | | | |
| Annex K | TCA eUICC Profile Package Test Specification | M | See Annex K |

Table 5: Applicability of Tests

| Conditional item | Condition |
| --- | --- |
| C001 | IF (O\_E\_NIST) THEN M ELSE N/A |
| C002 | IF (O\_E\_BRP) THEN M ELSE N/A |
| C006 | IF (NOT O\_E\_LPAe) THEN M ELSE N/A |
| C019 | IF (O\_E\_REUSE\_OTPK) THEN M ELSE N/A |
| C020 | IF (NOT O\_E\_REUSE\_OTPK) THEN M ELSE N/A |
| C025 | IF (O\_E\_2\_PIR) THEN M ELSE N/A |
| C032 | IF (O\_E\_CATBUSY AND NOT O\_E\_MEP) THEN M ELSE N/A |
| C033 | IF (NOT O\_E\_CATBUSY AND NOT O\_E\_MEP) THEN M ELSE N/A |
| C036 | IF (O\_E\_CATBUSY\_EN\_DIS\_NOREFRESH AND NOT O\_E\_MEP) THEN M ELSE N/A |
| C037 | IF (NOT O\_E\_CATBUSY\_EN\_DIS\_NOREFRESH AND NOT O\_E\_MEP) THEN M ELSE N/A |
| C038 | IF (O\_E\_CATBUSY\_MR AND NOT O\_E\_MEP) THEN M ELSE N/A |
| C039 | IF (NOT O\_E\_CATBUSY\_MR AND NOT O\_E\_MEP) THEN M ELSE N/A |
| C040 | IF O\_E\_INTEGRATED THEN M ELSE N/A |
| Conditions applicable to SGP.23 v3.1 onwards | |
| C301 | IF (O\_E\_RPM) THEN M ELSE N/A |
| C302 | IF (O\_E\_RPM AND O\_E\_ENTERPRISE) THEN M ELSE N/A |
| C303 | IF (O\_E\_RPM AND O\_E\_LPA\_PROXY) THEN M ELSE N/A |
| C304 | IF (O\_E\_RPM AND O\_E\_CAT\_BUSY) THEN M ELSE N/A |
| C305 | IF (O\_E\_RPM AND NOT O\_E\_MEP) THEN M ELSE N/A |
| C306 | IF (O\_E\_LPAe) THEN M ELSE N/A |
| C309 | IF (O\_E\_ENTERPRISE) THEN M ELSE N/A |
| C310 | IF (O\_VAR\_O) THEN M ELSE N/A |
| C311 | IF (O\_VAR\_OV3) THEN M ELSE N/A |
| C312 | IF (O\_VAR\_A) THEN M ELSE N/A |
| C313 | IF (O\_VAR\_B) THEN M ELSE N/A |
| C314 | IF (O\_VAR\_C) THEN M ELSE N/A |
| C315 | IF (O\_VAR\_A OR O\_VAR\_B OR O\_VAR\_C) THEN M ELSE N/A |
| C316 | IF ((O\_VAR\_A OR O\_VAR\_B OR O\_VAR\_C OR O\_VAR\_OV3) AND O\_E\_NIST) THEN M ELSE N/A |
| C317 | IF (O\_VAR\_A OR O\_VAR\_B OR O\_VAR\_C OR O\_VAR\_OV3) THEN M ELSE N/A |
| C318 | IF (O\_E\_MEP\_A1) THEN M ELSE N/A |
| C319 | IF (NOT O\_E\_MEP) THEN M ELSE N/A |
| C320 | IF (NOT O\_E\_CATBUSY\_EN\_DIS\_NOREFRESH) THEN M ELSE N/A |
| C321 | IF (NOT O\_E\_CATBUSY) THEN M ELSE N/A |
| C322 | IF (O\_E\_CATBUSY) THEN M ELSE N/A |
| C323 | IF (O\_E\_CATBUSY\_EN\_DIS\_NOREFRESH) THEN M ELSE N/A |
| C324 | IF (O\_E\_MEP\_A1 AND O\_E\_CATBUSY\_MR) THEN M ELSE N/A |
| C325 | IF O\_E\_LPAe THEN M ELSE N/A |
| C326 | IF (O\_E\_MEP\_A1 AND NOT O\_E\_CATBUSY\_MR) THEN M ELSE N/A |
| C327 | IF (O\_E\_MEP\_B) THEN M ELSE N/A |
| C328 | IF (O\_E\_MEP\_A2) THEN M ELSE N/A |
| C329 | IF (O\_E\_MEP\_A1 AND NOT O\_E\_CATBUSY\_EN\_DIS\_NOREFRESH) THEN M ELSE N/A |
| C330 | IF (O\_E\_MEP\_A2 AND O\_E\_CATBUSY\_EN\_DIS\_NOREFRESH) THEN M ELSE N/A |
| C331 | IF (O\_E\_MEP\_A2 AND NOT O\_E\_CATBUSY\_EN\_DIS\_NOREFRESH) THEN M ELSE N/A |
| C332 | IF (O\_E\_MEP\_A1 AND O\_E\_CATBUSY) THEN M ELSE N/A |
| C333 | IF (O\_E\_MEP\_A1 AND NOT O\_E\_CATBUSY) THEN M ELSE N/A |
| C334 | IF (O\_E\_MEP\_B AND O\_E\_CATBUSY) THEN M ELSE N/A |
| C335 | IF (O\_E\_MEP\_B AND NOT O\_E\_CATBUSY) THEN M ELSE N/A |
| C336 | IF (O\_E\_MEP\_A1 AND O\_E\_CATBUSY\_EN\_DIS\_NOREFRESH) THEN M ELSE N/A |
| C337 | IF (O\_E\_MEP\_B\_NO\_REFRESH AND NOT O\_E\_CATBUSY\_EN\_DIS\_NOREFRESH) THEN M ELSE N/A |
| C338 | IF (O\_E\_MEP\_B\_NO\_REFRESH AND O\_E\_CATBUSY\_EN\_DIS\_NOREFRESH) THEN M ELSE N/A |
| C339 | IF (O\_E\_MEP\_A2 AND O\_E\_CATBUSY\_MR) THEN M ELSE N/A |
| C340 | IF (O\_E\_MEP\_A2 AND NOT O\_E\_CATBUSY\_MR) THEN M ELSE N/A |
| C341 | IF (O\_E\_MEP\_B AND O\_E\_CATBUSY\_MR) THEN M ELSE N/A |
| C342 | IF (O\_E\_MEP\_B AND NOT O\_E\_CATBUSY\_MR) THEN M ELSE N/A |
| C345 | IF (O\_E\_MEP\_B\_NO\_REFRESH) THEN M ELSE N/A |
| C346 | IF (O\_E\_NON\_IMSI\_SUPI\_TYPE AND O\_E\_ADD\_PP\_VERSIONS AND O\_E\_DEVICE\_INFO\_EXTENSIBILITY\_SUPPORT) THEN M ELSE N/A |
| C347 | IF (O\_E\_DEVICE\_CHANGE) THEN M ELSE N/A |
| C348 | IF (O\_E\_HRI\_ADDRESS\_IN\_PM) THEN M ELSE N/A |
| C349 | IF (O\_E\_ENTERPRISE AND NOT O\_E\_MEP) THEN M ELSE N/A |
| C350 | IF (O\_E\_RPM AND O\_E\_LPA\_PROXY AND NOT O\_E\_MEP) THEN M ELSE N/A |
| C351 | IF (O\_E\_RPM AND O\_E\_ENTERPRISE AND NOT\_O\_E\_MEP) THEN M ELSE N/A |
| C352 | IF (O\_E\_DEVICE\_CHANGE AND NOT O\_E\_MEP) THEN M ELSE N/A |
| C353 | IF (O\_E\_LPA\_PROXY) THEN M ELSE N/A |
| C354 | IF (O\_E\_LPA\_PROXY AND NOT O\_E\_MEP) THEN M ELSE N/A |
| C355 | IF (O\_E\_HRI\_ADDRESS\_IN\_PM AND NOT O\_E\_MEP) THEN M ELSE N/A |
| C356 | IF (NOT O\_E\_ENTERPRISE) THEN M ELSE N/A |
| C357 | IF (O\_VAR\_O and O\_E\_NIST) THEN M ELSE N/A |
| C358 | IF (O\_VAR\_O and O\_E\_BRP) THEN M ELSE N/A |

Table 6: Conditional Items Referenced by Table 5

Note: Conditions C0XX which are missing in Table 6 are present in an earlier version of SGP.23 but are not used in the current version. Conditions RFU are for future use in the next release of this specification.

## 2.2 General Consideration

This section contains some general considerations about the test cases defined in this document. Note that some external test specifications are referred to in chapter 7. Consequently, the following sub sections SHALL only apply for test cases defined in sections 4 and 5 and 6.

### 2.2.1 Test Case Definition

Test descriptions are independent.

For each test described in this document, a chapter provides a general description of the initial conditions applicable for the whole test. This description is completed by specific configurations to each individual sub-case.

It is implicitly assumed that the IUT SHALL be compliant with the initial states described in Annex G. An initial state SHALL be considered as a pre-requisite to execute all the test cases described in this Test Plan.

After completing the test, the configuration is reset before the execution of the following test.

### 2.2.2 Test Cases Format

Here is an explanation of the way to define the test cases in chapters 4, 5 and 6.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **4.X.Y.Z Test Cases**  **4.X.Y.Z.1 TC\_IUT\_TestName1**   |  |  | | --- | --- | | General Initial Conditions | | | Entity | Description of the general initial condition | | Entity1 | Test case - general condition 1 | | Entity2 | Test case - general condition 2 |   **Test Sequence #01: Short Description**  Description of the aim of the test sequence N°1   |  |  | | --- | --- | | Initial Conditions |  | | Entity | Description of the initial condition | | Entity1 | Test sequence N°1 - initial condition 1 | | Entity2 | Test sequence N°1 - initial condition 2 |  | Step | Direction | Sequence / Description | Expected result | | --- | --- | --- | --- | | IC1 | Entity1 → Entity2 | Command or Message to send from Entity1 to Entity2 | Expected result N°1.1 | | 1 | Entity1 → Entity2 | Command or Message to send from Entity1 to Entity2 | 1- expected result N°1.2  2- expected result N°1.3 | | 2 | Entity2 → Entity3 | Command or Message to send from Entity2 to Entity3 |  |   **Test Sequence #02**  Description of the aim of the test sequence N°2   | Step | Direction | Sequence / Description | Expected result | | --- | --- | --- | --- | | 1 | Entity1 → Entity2 | Command or Message to send from Entity1 to Entity2 |  | | 2 | Entity2 → Entity3 | Command or Message to send from Entity2 to Entity3 | 1- expected result N°2.1  2- expected result N°2.2 |   **4.X.Y.Z.2 TC\_IUT\_TestName2**  … |

The test cases TC\_IUT\_TestName1 and TC\_IUT\_TestName2 are referenced in Table 5 that allows indicating the applicability of the tests.

In the test case TC\_IUT\_TestName1, the requirements REQ1 and REQ2 are respectively covered by the test sequences #01 and #02.

The test sequence #01 SHALL be executed if and only if these conditions are met:

* Test case - general condition 1
* Test case - general condition 2
* Test sequence N°1 - initial condition 1
* Test sequence N°1 - initial condition 2

The test sequence #02 SHALL be executed if and only if these conditions are met:

* Test case - general condition 1
* Test case - general condition 2

The tables defining the different initial conditions are optional.

Initial Conditions are intended to be reached dynamically using the Test Tool when possible.

No additional operation SHALL be done prior to the test sequence besides those indicated in the Initial Conditions (e.g. no other Profiles SHALL be present on the eUICC besides those defined in the Initial Conditions).

In the test sequence #01:

* the step IC1 corresponds to an additional Initial Condition
* in the step N°1, if the expected results N°1 and N°2 are validated, the requirement REQ1 (or a part of the REQ1) SHALL be considered as implemented

Note that all initial states (described in Annex G) SHALL be implemented by the IUT whatever the test cases to execute.

In addition, following 2.2.2 sub sections present all information (e.g. Methods, Constants…) that MAY be referenced in test sequences.

After execution of each test sequence a clean-up procedure (CU) SHALL be executed to restore the IUT to the Common Initial State as defined in Annex G.

Sections 2.2.2.0 to 2.2.2.6 specify how to factor text among test sequences.

**2.2.2.0 Conditional Execution of Test Sequence Steps**

Some steps may be relevant only if the IUT supports the corresponding option, even if the intent and general script of the enclosing Test Sequence is applicable to several options.

In that case the Test Sequence uses the following construct:

| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| --- | --- | --- | --- |
| 1 | Entity1 → Entity2 | Command or Message to send from Entity1 to Entity2 | Expected result N°1.1 |
| IF (optionsCombination) | | | |
| 2 | Entity1 → Entity2 | Command or Message to send from Entity1 to Entity2 | 1- expected result N°2.1  2- expected result N°2.2 |
| 3 | Entity2 → Entity3 | Command or Message to send from Entity2 to Entity3 |  |
| ENDIF | | | |
| 4 | Entity1 → Entity2 | Command or Message to send from Entity1 to Entity2 |  |

The test tool SHALL execute the steps enclosed between keywords IF and ENDIF if and only if the expression optionsCombination has value ‘true’.

Every IF is closed by the first ENDIF that comes after it. Nesting is not supported.

The expression optionsCombination is a combination of one or more Boolean values that can be:

* options defined in section 2.1.4,
* comparisons on the value of IUT\_SETTINGS
* comparisons on the value of dynamic content defined in see Annex B

and using the boolean operators OR, AND, NOT, with brackets to avoid any ambiguity on operator precedence.

#### 2.2.2.1 Methods and Procedures

A method factors the computation of a single value, based on parameters passed by a test step in a test sequence.

A method is referenced as follow:

 MTD\_NAME\_OF\_THE\_METHOD(PARAM1, PARAM2…)

The key word “NO\_PARAM” SHALL be set in method call if the related optional parameter is not used.

All methods and their related parameters are described in Annex C.1.

A procedure factors a generic sub-sequence of test steps. It may take parameters to exert these test steps on different contexts or with different values.

A procedure is referenced as follow:

* PROC\_NAME\_OF\_THE\_PROCEDURE if it doesn't take parameters
* PROC\_NAME\_OF\_THE\_PROCEDURE(PARAM1, PARAM2…) if it takes parameters. In that case, the key word “NO\_PARAM” SHALL be set if the related optional parameter is not used.

All procedures are described in Annex C.2.

The implementation of these methods and procedures is under the responsibility of the test tool providers.

#### 2.2.2.2 Constants and Dynamic Content

A constant (e.g. text, ASN.1 structure, hexadecimal string, icon, URI, integer, EID, AID…) is referenced as follow:

 #NAME\_OF\_THE\_CONSTANT

All constants are defined in Annex A.

When provided as an ASN.1 value notation, a constant SHALL be encoded in DER TLV (as specified in ITU-T X.690 [16]) by the test tool.

A dynamic content (e.g. TLV, ASN.1 structure, signature, integer, AID, one-time key pair…) is referenced as follow:

 <NAME\_OF\_THE\_VARIABLE>

All dynamic contents are defined in Annex B.

A dynamic content is either generated by an IUT or by a test tool provider.

#### 2.2.2.3 Requests and Responses

An ASN.1 or a JSON request is referenced as follow:

 #NAME\_OF\_THE\_REQUEST

An ASN.1 or a JSON response is referenced as follows:

 #R\_NAME\_OF\_THE\_RESPONSE

Each ASN.1 or JSON request and response MAY refer to a constant or a dynamic content. All these structures are defined in Annex D.

When provided as an ASN.1 value notation, a request or a response SHALL be encoded in DER TLV (as specified in ITU-T X.690 [16]) by the test tool.

When an ASN.1 element definition contains three points (i.e. “…”), it means that fields MAY be present but SHALL not be checked by the test tool.

In the following example, several fields MAY be part of the ProfileInfoListResponse but only the profileNickname SHALL be verified.

resp ProfileInfoListResponse ::=

profileInfoListOk :{

{  
 ...  
 profileNickname #NICKNAME  
 ...

}

}

This rule applies aslo for Constants definition.

Some ASN.1 SEQUENCE components have a DEFAULT value (for example, profileClass in StoreMetadataRequest). In this specification, when values are specified in ASN.1 syntax and the DEFAULT value is intended, two different formulations (both of which are valid) may be used:

* the relevant component is specified with the DEFAULT value;
* the relevant component is missing entirely.

These are logically equivalent and lead to the same DER encoding. In both cases, the following rules apply:

* When the test tool is sending the DER value, it SHALL NOT include the component (as per DER rules).
* When the test tool is checking a received DER value from the entity under test, it SHALL check that the component is NOT present.

Test tools SHALL consider two BIT STRINGs to be equivalent if the BIT STRINGs have the same DER encoding. For example, '0101'B shall be considered to be equivalent to '010100'B.

NOTE: this is equivalent to removing any trailing zero bits from the BIT STRINGs in "bstring" notation (e.g. '010100'B 🡪 '0101'B) and then comparing the strings textually.

NOTE: according to the DER format, the encoding of transmitted values will remove the trailing zeroes. The definition above allows for values which are specified using ASN.1 value notation and are not transmitted, such as values specified in the Annexes of the current document, including IUT settings which might be specified by a user of the current document and may contain trailing zeroes in the ASN.1 value notation.

#### 2.2.2.4 APDUs

A C-APDU is referenced as follow:

 [NAME\_OF\_THE\_CAPDU]

All C-APDUs are defined in Annex D.4, except the APDU STORE DATA which is constructed by one of the methods MTD\_STORE\_DATA\_xxx, see annex C.1.

An R-APDU is referenced as follow:

[R\_NAME\_OF\_THE\_RAPDU]

All R-APDUs are defined in Annex D.4.

Each APDU MAY refer to a constant or a dynamic content.

The APDU TERMINAL RESPONSE SHALL be dynamically generated by the test tool according to the related proactive command. Therefore, this particular command is not referenced with brackets in this specification. If not explicitly defined in the step, the general result SHALL be set by default to “Command performed successfully” (i.e. 0x83 01 00).

#### 2.2.2.5 Profiles

In order to execute the test cases described in this document, Operational, Test and Provisioning Profiles are necessary. All these Profiles are defined in Annex E with the Profile Metadata content and the corresponding Profile Package as defined in the eUICC Profile Package Specification [4].

A Profile is referenced as follow:

 PROFILE\_OPERATIONALx with x the identifier of the Operational Profile

or

 PROFILE\_TESTx with x the identifier of the Test Profile

or

 PROFILE\_PROVISIONINGx with x the identifier of the Provisioning Profile

NOTE: Test Profiles and Provisioning Profiles are out of the scope of this version of test specification.

#### 2.2.2.6 IUT Settings

For the purpose of some test cases, eUICC manufacturers need to give some information related to their products to the test tools providers (e.g. supported Java Card version).

An IUT setting is referenced as follow:

 #IUT\_NAME\_OF\_SETTING

All these settings are defined in Annex F.

#### 2.2.2.7 Referenced Requirements

All requirements referenced in this document by their identifiers are present and described in Annex I. These requirements have been extracted from the specifications:

 GSMA RSP Technical Specification [2]

 GSMA RSP Architecture [3]

### 2.2.3 General Rules for eUICC Testing

#### 2.2.3.1 Default Profile Downloading process

By default, when an Operational Profile needs to be downloaded on the eUICC (e.g. as mentioned in an initial condition), the following rules apply except if it is differently defined in the Test Case.

In order to execute the Common Mutual Authentication procedure and the Sub-procedure Profile Download and Installation (End User Confirmation), the following requests SHALL be sent by the Test Tool:

 #GET\_EUICC\_INFO1 and #GET\_EUICC\_CHALLENGE

 MTD\_AUTHENTICATE\_SMDP

o o with the SM-DP+ address #TEST\_DP\_ADDRESS1

o with #CTX\_PARAMS1\_MATCH\_ID

o If euiccCiPKIdListForSigningV3 is present in eUICCInfo1, select the eSIM CA RootCA Public Key, following the priority order given by the eUICC in euiccCIPKIdListForSigningV3. The paramNewCertVariant SHALL be set to TRUE.

o If euiccCiPKIdListForSigningV3 is not present in eUICCInfo1, select the eSIM CA RootCA Public Key, following the priority order given by the eUICC in euiccCIPKIdListForSigning. The paramNewCertVariant SHALL be set to FALSE.

o with the #CERT\_S\_SM\_DPauth\_SIG leading to the same CI as the one chosen for signing

o with the #CRL\_LIST included

 #PREP\_DOWNLOAD\_NO\_CC

o with the #CERT\_S\_SM\_DPpb\_SIG leading to the same CI as the one chosen for signing

 Neither ES10b.GetRAT nor ES10b.GetProfilesInfo requests SHALL be executed

During the Profile Installation, the following SCP03t TLVs SHALL be used by default:

 #S\_INIT\_SC\_PROF1

 #CONF\_ISDP\_EMPTY

 no TLV for "ES8+.ReplaceSessionKeys" function SHALL be used (i.e. the Profile SHALL be downloaded by using the session keys <S\_ENC> and <S\_MAC>)

#### 2.2.3.2 Default Local Profile Management process

By default, when an Operational Profile needs to be enabled, disabled or deleted on the eUICC (e.g. As mentioned in an initial condition), the following rules apply except if it is differently defined in the Test Case.

If ‘LSI Support’ is not included in the ATR:

The EnableProfileRequest and the DisableProfileRequest SHALL contain the following parameters:

 ICCID of the Profile to Enable or to Disable

 RefreshFlag set to TRUE

The eUICC SHALL send the REFRESH command in "UICC Reset" mode (i.e. the APDU[TERMINAL\_PROFILE] indicating the support "UICC Reset" SHALL be used by the Test Tool).

If ‘LSI Support’ is included in the ATR:

The test tool SHALL confirm ‘LSI Support’ in the PPS2 procedure and SHALL send MANAGE\_LSI(Configure LSI) with the following tags:

 ‘Highest LSI proposed by the terminal’ set to 3.

 ‘LSI options supported by the terminal’ set to 0x01.

 ‘MEP mode(s) supported by the S\_Device in the order of priority’ set to (unless otherwise specified by the Test Sequence):

'01': MEP-A1,

‘02’: MEP-A2,

‘03’: MEP-B.

 ‘Maximum number of LSIs for Enabled Profiles of the Device’ set to 2.

Test tool SHALL process the response to MANAGE\_LSI(Configure LSI) as follow.

* Accept the ‘LSI options’ returned by the eUICC. If it is absent, consider it as 0x00.
* Select the MEP mode, <MEP\_MODE>, to be used as the ‘Jointly supported MEP mode’.

Test tool SHALL select eSIM Port=0 as the command port.

Test tool SHALL indicate support of LSI COMMAND with "Proactive Session  
Request" and Proactive UICC: LSI COMMAND with "UICC Platform Reset”.

If <MEP\_MODE> is MEP-A1:

The EnableProfileRequest and the DisableProfileRequest SHALL contain the following parameters:

 ICCID of the Profile to Enable or to Disable

 RefreshFlag set to TRUE

 targetEsimPort set to 1 or 2

The eUICC SHALL send the LSI COMMAND with the action “Proactive session request” with the targetEsimPort on the eSIM Port 0. The test tool SHALL FETCH the LSI COMMAND on port 0, send the TERMINAL RESPONSE on the same port, and then FETCH the REFRESH command on the targetEsimPort.

The eUICC SHALL send the REFRESH command in "eUICC Profile State Change" mode or the “UICC Reset” mode (i.e. the APDU[TERMINAL\_PROFILE] indicating the support "eUICC Profile State Change" or “UICC Reset” SHALL be used by the Test Tool) on the targetEsimPort.

If the REFRESH mode is "eUICC Profile State Change", the Test Tool SHALL send the TERMINAL RESPONSE on the targetEsimPort. If the REFRESH mode is "UICC Reset", Test Tool SHALL send the MANAGE\_LSI(Reset LSE) on the targetEsimPort.

If <MEP\_MODE> is MEP-A2:

The EnableProfileRequest and the DisableProfileRequest SHALL contain the following parameters:

 ICCID of the Profile to Enable or to Disable

 RefreshFlag set to FALSE

The test tool SHALL not send the targetEsimPort, but SHALL extract and verify the targetEsimPort returned by the eUICC in the EnableProfileResponse.

If <MEP\_MODE> is MEP-B:

The EnableProfileRequest and the DisableProfileRequest SHALL contain the following parameters:

 ICCID of the Profile to Enable or to Disable

 RefreshFlag set to TRUE

The test tool SHALL not send the targetEsimPort as it is the same as the command port.

The test tool SHALL FETCH the REFRESH command on the command port.

The eUICC SHALL send the REFRESH command in "eUICC Profile State Change" mode or the “UICC Reset” mode (i.e. the APDU[TERMINAL\_PROFILE] indicating the support "eUICC Profile State Change" or “UICC Reset” SHALL be used by the Test Tool) on the command port.

If the REFRESH mode is "eUICC Profile State Change", the Test Tool SHALL send the TERMINAL RESPONSE on the same command port. If the REFRESH mode is "UICC Reset", Test Tool SHALL send the MANAGE\_LSI(Reset LSE) on the same command port.

The DeleteProfileRequest SHALL contain the following parameter:

 ICCID of the Profile to Delete

#### 2.2.3.3 ASN.1 elements verifications

Each time the eUICC returns an ASN.1 structure containing a SEQUENCE OF elements, the order of elements SHALL be checked by the Test Tool except for the particular responses:

 notificationMetadataList of ListNotificationResponse

 profileInfoListOk of ProfileInfoListResponse

 notificationList of RetrieveNotificationsListResponse

#### 2.2.3.4 Default Remote Profile Management process

By default, when an Operational Profile needs to be enabled, disabled or deleted on the eUICC by using an RPM script (e.g. as mentioned in an initial condition), the following rules apply except if it is differently defined in the Test Case.

In order to execute the Common Mutual Authentication procedure, the following requests SHALL be sent by the Test Tool:

 #GET\_EUICC\_INFO1 and #GET\_EUICC\_CHALLENGE

 MTD\_AUTHENTICATE\_SMDP

* + - * with the SM-DP+ address #TEST\_DP\_ADDRESS1
      * with #CTX\_PARAMS1\_RPM
      * If euiccCiPKIdListForSigningV3 is present in eUICCInfo1, select the eSIM CA RootCA Public Key, following the priority order given by the eUICC in euiccCIPKIdListForSigningV3. The paramNewCertVariant SHALL be set to TRUE.
      * If euiccCiPKIdListForSigningV3 is not present in eUICCInfo1, select the eSIM CA RootCA Public Key, following the priority order given by the eUICC in euiccCIPKIdListForSigning. The paramNewCertVariant SHALL be set to FALSE.
      * with the #CERT\_S\_SM\_DPauth\_SIG leading to the same CI as the one chosen for signing
      * with the #CRL\_LIST included

No ES10b.GetProfilesInfo request SHALL be executed.

In order to execute the Sub-procedure RPM Execution, the LoadRpmPackageRequest SHALL contain the following parameters:

* only one RpmCommand (corresponding to the requested operation)
* the ICCID of the Profile to Enable, to Disable or to Delete
* the parameters continueOnFailure and rpmPending SHALL not be set

In case the RPM script aims to Disable or Enable a Profile, the eUICC SHALL send the REFRESH command in "UICC Reset" mode (i.e. the APDU [TERMINAL\_PROFILE] indicating the support "UICC Reset" SHALL be used by the Test Tool).

#### 2.2.3.5 Default Common Mutual Authentication procedure

By default, when Common Mutual Authentication procedure has to be executed only as a pre-step (e.g. as mentioned in an initial condition for the test cases where only some particular steps of the Profile Download are verified), the following rules apply except if it is differently defined in the Test Case.

The following requests SHALL be sent by the Test Tool:

 #GET\_EUICC\_INFO1 and #GET\_EUICC\_CHALLENGE

 MTD\_AUTHENTICATE\_SMDP

o with the SM-DP+ address #TEST\_DP\_ADDRESS1

o with #CTX\_PARAMS1\_MATCH\_ID

o If euiccCiPKIdListForSigningV3 is present in eUICCInfo1, select the eSIM CA RootCA Public Key, following the priority order given by the eUICC in euiccCIPKIdListForSigningV3. The paramNewCertVariant SHALL be set to TRUE.

* + If euiccCiPKIdListForSigningV3 is not present in eUICCInfo1, select the eSIM CA RootCA Public Key, following the priority order given by the eUICC in euiccCIPKIdListForSigning. The paramNewCertVariant SHALL be set to FALSE.

o with the #CERT\_S\_SM\_DPauth\_SIG leading to the same CI as the one chosen for signing

o with the #CRL\_LIST included

#### 2.2.3.6 Usage of Logical Channels

All ES10 commands are wrapped in a STORE DATA APDU, and sent in a Logical Channel dedicated to sending commands to the ISD-R.

All other APDus (in particular, APDUs specified in Annex D.4) are sent in the Basic Logical Channel.

### 2.2.4 VOID

### 2.2.5 Pass Criteria

A test execution is considered as successful only if the test procedure was fully carried out successfully.

A test execution is considered as failed if the tested feature provides an unexpected behaviour.

A test execution is considered as inconclusive when the pass criteria cannot be evaluated due to issues during the setup of the initial conditions (including the ICx steps) or during the execution of steps in which no requirement is referenced.

### 2.2.6 Future Study

Some of the test cases or test sequences described in this Test Plan are FFS (For Future Study). This MAY mean that some clarifications are expected at the requirement level to conclude on a test method. As consequence, the corresponding test SHALL not be executed.

# 3 Testing Architecture

## 3.1 Testing Scope

All the interfaces, intended to be tested in the scope of this document, are presented hereafter:



| Interface | Between | | Description |
| --- | --- | --- | --- |
| ES6 | Operator | eUICC | Used by the Operator for the management of Operator services via OTA services. |
| ES8+ | SM-DP+ | eUICC | Provides a secure end-to-end channel between the SM-DP+ and the eUICC for the administration of the ISD-P and the associated Profile during download and installation. It provides Perfect Forward Secrecy. |
| ES10a | LDSd | eUICC | Used between the LDSd and the LPA Services to handle a Profile discovery. |
| ES10b | LPDd | eUICC | Used between the LPDd and the LPA services to transfer a Bound Profile Package to the eUICC. This interface plays no role in the decryption of Profile Packages. |
| ES10c | LUId | eUICC | Used between the LUId and the LPA services for Local Profile Management by the End User. |

Table 7: Tested Interfaces Descriptions

## 3.2 Testing Execution

This chapter aims to describe the different testing environments and equipments to allow the test cases to be executed.

To permit the execution of the different test cases described in this Test Plan, specifics simulators SHALL be used. The simulators that have been defined are listed hereafter:

 S\_Device: the Device Simulator used to send some commands to the eUICC under test using ISO/IEC 7816-4 [7] on the contact interface

 S\_SM-DP+: the SM-DP+ Simulator

 S\_SM-DS: the SM-DS Simulator

 S\_MNO: the MNO Simulator

 S\_LPAd: the LPAd Simulator

Implementation of these simulators remains under the responsibility of the test tool providers. The aim of all the test cases is to verify the compliance of an Actor/Component (i.e. eUICC).

Following notations are used:

 S\_ComponentName for a simulated component

 ComponentName for the Implementation Under Test (IUT)

The use of "-- optional" in any ASN.1 elements defined within this document indicate that the test tool SHALL allow for the value either being present with that value, or being absent.

### 3.2.1 eUICC - Test Environment

The following test environment is used for all eUICC test cases as defined in chapter 4.2 and 5.2 (unless it is specified differently in the specific test case). Following conditions apply:

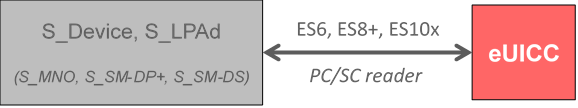
 Removable eUICC is used

 EUM SHALL provide products with one of the form factors specified in ETSI TS 102 221 [5]

 EUM SHALL provide products compliant with Annex G.2 – eUICC Initial States

 LPAd / MNO / SM-DP+ / SM-DS / Device Simulators SHALL be implemented by the test tools

 The Device Simulator SHALL support both LSI indication options (*MANAGE LSI(Select LSI)* and *T=1 using NAD byte*) allowed for MEP, and use the one indicated by the EUM in IUT\_EUICC\_MULTIPLEXING\_LSI\_INDICATION.



The reference of this Test Environment is TE\_eUICC.

### 3.2.2 VOID

### 3.2.3 VOID

### 3.2.4 VOID

### 3.2.5 Integrated eUICC - Test Environment

The following test environment is used for all eUICC test cases as defined in chapter 4.2 and 5.2 (unless it is specified differently in the specific test case). Following conditions apply:

* EUM SHALL provide products compliant with Annex G.2 – eUICC Initial States
* LPAd / MNO / SM-DP+ / SM-DS / Device Simulators SHALL be implemented by the test tools
  + The Device Simulator SHALL support the same commands, responses, and functionalities as in section 3.2.1, and differs only by the support of a dedicated physical interface.
* Integrated eUICC shall provide a test interface which includes one of the following:
  + ISO/IEC 7816-4 [7]
  + USB CCID [28]
* For Integrated eUICC providing a USB CCID [28] test interface, the provisions of Annex J SHALL apply
* For Integrated eUICC providing ISO/IEC 7816-4 [7], the requirements of 3.2.1 eUICC - Test Environment with implementing shall apply



The reference of this [28] USB CCID based Test Environment is TE\_Integrated eUICC.

# 4 Interface Compliance Testing

## 4.1 General Overview

This section focuses on the implementation of the different interfaces according to the GSMA RSP Technical Specification [2]. The aim is to verify the compliance of all eUICC interfaces.

## 4.2 eUICC Interfaces

### 4.2.1 ATR and ISD-R Selection

#### 4.2.1.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 3.4.1
* Section 5.7
* Annex D

#### 4.2.1.2 Test Cases

##### 4.2.1.2.1 TC\_eUICC\_ATR\_And\_ISDR\_Selection

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is loaded on the eUICC and, unless specified otherwise by the test sequence, is in Disabled state. |

Test Sequence #01 Nominal: ATR and Select ISD-R

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_Device → eUICC | RESET | ATR present with the first tBi (i>2) after T = 15 containing b2=1 |
| 2 | S\_Device → eUICC | [SELECT\_MF] | FCP Template present  SW=0x9000 |
| 3 | S\_Device → eUICC | [TERMINAL\_CAPABILITY\_LPAd] | SW=0x9000 |
| 4 | S\_Device → eUICC | [TERMINAL\_PROFILE] | Toolkit initialization THEN SW=0x9000 |
| 5 | S\_LPAd → eUICC | [MANAGE\_CHANNEL\_OPEN] | Extract the <CHANNEL\_NUMBER> from response data  SW=0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_SELECT(#ISD\_R\_AID) | The response data:  0x6F <L>   84 <L> #ISD\_R\_AID   A5 <L> <PROPRIETARY\_DATA>  #R\_ISDR\_SELECTION  SW=0x9000 |

Test Sequence #02 Nominal: ATR and Select ISD-R with Enabled Profile

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_Device → eUICC | RESET | ATR present with the first tBi (i>2) after T = 15 containing b2=1 |
| 2 | S\_Device → eUICC | [SELECT\_MF] | FCP Template present  SW=0x9000 |
| 3 | S\_Device → eUICC | [TERMINAL\_CAPABILITY\_LPAd] | SW=0x9000 |
| 4 | S\_Device → eUICC | [TERMINAL\_PROFILE] | Toolkit initialization THEN SW=0x9000 |
| 5 | S\_LPAd → eUICC | [MANAGE\_CHANNEL\_OPEN] | Extract the <CHANNEL\_NUMBER> from response data  SW=0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_SELECT(#ISD\_R\_AID) | The response data:  0x6F <L>   84 <L> #ISD\_R\_AID   A5 <L> <PROPRIETARY\_DATA> #R\_ISDR\_SELECTION\_EN\_PROF  SW=0x9000 |

Test Sequence #03 Nominal: ATR and Select ISD-R when LPAe supported

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_Device → eUICC | RESET | ATR present with the first tBi (i>2) after T = 15 containing b2=1 |
| 2 | S\_Device → eUICC | [SELECT\_MF] | FCP Template present  SW=0x9000 |
| 3 | S\_Device → eUICC | [TERMINAL\_CAPABILITY\_LPAd] | SW=0x9000 |
| 4 | S\_Device → eUICC | [TERMINAL\_PROFILE] | Toolkit initialization THEN SW=0x9000 |
| 5 | S\_LPAd → eUICC | [MANAGE\_CHANNEL\_OPEN] | Extract the <CHANNEL\_NUMBER> from response data  SW=0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_SELECT(#ISD\_R\_AID) | The response data:  0x6F <L>   84 <L> #ISD\_R\_AID   A5 <L> <PROPRIETARY\_DATA>  #R\_ISDR\_SELECTION\_LPAE  SW=0x9000 |

Test Sequence #04 Nominal: ATR and Select ISD-R for MEP-A1

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| 2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify <MEP\_MODE> = ’01’,  Verify <MEP\_LSI\_OPTION> =  #IUT\_MEP\_LSI\_OPTIONS,  Verify <MEP\_MAX\_LSIS> <=  #IUT\_MEP\_MAX\_LSIS |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 4 | S\_Device → eUICC | [SELECT\_MF] | FCP Template present  SW=0x9000 |
| 5 | S\_Device → eUICC | [TERMINAL\_CAPABILITY\_LPAd] | SW=0x9000 |
| 6 | S\_Device → eUICC | [TERMINAL\_PROFILE\_LSI\_COMMAND] | Toolkit initialization THEN SW=0x9000 |
| 7 | S\_LPAd → eUICC | [MANAGE\_CHANNEL\_OPEN] | Extract the <CHANNEL\_NUMBER> from response data  SW=0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_SELECT(#ISD\_R\_AID) | The response data:  0x6F <L>   84 <L> #ISD\_R\_AID   A5 <L> <PROPRIETARY\_DATA>  #R\_ISDR\_SELECTION  SW=0x9000 |

Test Sequence #05 Nominal: ATR and Select ISD-R for MEP-A2

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| 2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify <MEP\_MODE> = ’02’,  Verify <MEP\_LSI\_OPTION> =  #IUT\_MEP\_LSI\_OPTIONS,  Verify <MEP\_MAX\_LSIS> <=  #IUT\_MEP\_MAX\_LSIS |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 4 | S\_Device → eUICC | [SELECT\_MF] | FCP Template present  SW=0x9000 |
| 5 | S\_Device → eUICC | [TERMINAL\_CAPABILITY\_LPAd] | SW=0x9000 |
| 6 | S\_Device → eUICC | [TERMINAL\_PROFILE] | Toolkit initialization THEN SW=0x9000 |
| 7 | S\_LPAd → eUICC | [MANAGE\_CHANNEL\_OPEN] | Extract the <CHANNEL\_NUMBER> from response data  SW=0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_SELECT(#ISD\_R\_AID) | The response data:  0x6F <L>   84 <L> #ISD\_R\_AID   A5 <L> <PROPRIETARY\_DATA>  #R\_ISDR\_SELECTION  SW=0x9000 |

Test Sequence #06 Nominal: ATR and Select ISD-R for MEP-B

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| 2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030102”,  2) | Verify <MEP\_MODE> = ’03’,  Verify <MEP\_LSI\_OPTION> =  #IUT\_MEP\_LSI\_OPTIONS,  Verify <MEP\_MAX\_LSIS> <=  #IUT\_MEP\_MAX\_LSIS |
| 3 | S\_Device → eUICC | [SELECT\_MF] | FCP Template present  SW=0x9000 |
| 4 | S\_Device → eUICC | [TERMINAL\_CAPABILITY\_LPAd] | SW=0x9000 |
| 5 | S\_Device → eUICC | [TERMINAL\_PROFILE] | Toolkit initialization THEN SW=0x9000 |
| 6 | S\_LPAd → eUICC | [MANAGE\_CHANNEL\_OPEN] | Extract the <CHANNEL\_NUMBER> from response data  SW=0x9000 |
| 7 | S\_LPAd → eUICC | MTD\_SELECT(#ISD\_R\_AID) | The response data:  0x6F <L>   84 <L> #ISD\_R\_AID   A5 <L> <PROPRIETARY\_DATA>  #R\_ISDR\_SELECTION  SW=0x9000 |

Test Sequence #07 Nominal: ATR and Select ISD-R with Enabled Profile for MEP-A1

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on Port 1. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| 2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify <MEP\_MODE> = ’01’,  Verify <MEP\_LSI\_OPTION> =  #IUT\_MEP\_LSI\_OPTIONS,  Verify <MEP\_MAX\_LSIS> <=  #IUT\_MEP\_MAX\_LSIS |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 4 | S\_Device → eUICC | [SELECT\_MF] | FCP Template present  SW=0x9000 |
| 5 | S\_Device → eUICC | [TERMINAL\_CAPABILITY\_LPAd] | SW=0x9000 |
| 6 | S\_Device → eUICC | [TERMINAL\_PROFILE\_LSI\_COMMAND] | Toolkit initialization THEN SW=0x9000 |
| 7 | S\_LPAd → eUICC | [MANAGE\_CHANNEL\_OPEN] | Extract the <CHANNEL\_NUMBER> from response data  SW=0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_SELECT(#ISD\_R\_AID) | The response data:  0x6F <L>   84 <L> #ISD\_R\_AID   A5 <L> <PROPRIETARY\_DATA> #R\_ISDR\_SELECTION\_EN\_PROF  SW=0x9000 |

Test Sequence #08 Nominal: ATR and Select ISD-R with Enabled Profile for MEP-A2

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| 2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify <MEP\_MODE> = ’02’,  Verify <MEP\_LSI\_OPTION> =  #IUT\_MEP\_LSI\_OPTIONS,  Verify <MEP\_MAX\_LSIS> <=  #IUT\_MEP\_MAX\_LSIS |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 4 | S\_Device → eUICC | [SELECT\_MF] | FCP Template present  SW=0x9000 |
| 5 | S\_Device → eUICC | [TERMINAL\_CAPABILITY\_LPAd] | SW=0x9000 |
| 6 | S\_Device → eUICC | [TERMINAL\_PROFILE] | Toolkit initialization THEN SW=0x9000 |
| 7 | S\_LPAd → eUICC | [MANAGE\_CHANNEL\_OPEN] | Extract the <CHANNEL\_NUMBER> from response data  SW=0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_SELECT(#ISD\_R\_AID) | The response data:  0x6F <L>   84 <L> #ISD\_R\_AID   A5 <L> <PROPRIETARY\_DATA> #R\_ISDR\_SELECTION\_EN\_PROF  SW=0x9000 |

Test Sequence #09 Nominal: ATR and Select ISD-R with Enabled Profile for MEP-B

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on Port 1. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| 2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030102”,  2) | Verify <MEP\_MODE> = ’03’,  Verify <MEP\_LSI\_OPTION> =  #IUT\_MEP\_LSI\_OPTIONS,  Verify <MEP\_MAX\_LSIS> <=  #IUT\_MEP\_MAX\_LSIS |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 4 | S\_Device → eUICC | [SELECT\_MF] | FCP Template present  SW=0x9000 |
| 5 | S\_Device → eUICC | [TERMINAL\_CAPABILITY\_LPAd] | SW=0x9000 |
| 6 | S\_Device → eUICC | [TERMINAL\_PROFILE] | Toolkit initialization THEN SW=0x9000 |
| 7 | S\_LPAd → eUICC | [MANAGE\_CHANNEL\_OPEN] | Extract the <CHANNEL\_NUMBER> from response data  SW=0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_SELECT(#ISD\_R\_AID) | The response data:  0x6F <L>   84 <L> #ISD\_R\_AID   A5 <L> <PROPRIETARY\_DATA> #R\_ISDR\_SELECTION\_EN\_PROF  SW=0x9000 |

### 4.2.2 ES6 (Operator -- eUICC): UpdateMetadata

#### 4.2.2.1 Conformance Requirements

**References**

3GPP TS 23.040 - Technical realization of the Short Message Service (SMS) [22]

GSMA RSP Technical Specification [2]:

* Section 2.4.5
* Section 2.9.1, 2.9.3.2
* Section 5.4
* Section 5.7.15

#### 4.2.2.2 Test Cases

##### 4.2.2.2.1 TC\_eUICC\_ES6.UpdateMetadata

Throughout all the ES6.UpdateMetadata test cases, SMS is used as the secure OTA channel.

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_WITH\_PPRS\_AND\_ICON is loaded on the eUICC. |

Test Sequence #01 Nominal: Unset PPR1

The purpose of this test is to verify that the MNO can unset PPR1 from a Profile and that the eUICC can handle an Update Metadata request with only one field present.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| 1 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [INSTALL\_PERSO\_RES\_ISDP];  MTD\_STORE\_DATA\_SCRIPT(  #REMOVE\_PPR1,  FALSE)) | SW=0x91XX |
| 2 | S\_Device → eUICC | FETCH “XX” | MTD\_CHECK\_SMS\_POR(  0x9000) |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_NEW\_METADATA\_V3) | #R\_GET\_UPDATE\_N1  SW=0x9000 |

Test Sequence #02 Nominal: Unset PPR2 and update icon

The purpose of this test is to verify that the MNO can unset PPR2 and update the icon and icon type values from a Profile.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| 1 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [INSTALL\_PERSO\_RES\_ISDP];  MTD\_STORE\_DATA\_SCRIPT(  #UPD\_ICON\_REM\_PPR2,  FALSE)) | SW=0x91XX |
| 2 | S\_Device → eUICC | FETCH “XX” | MTD\_CHECK\_SMS\_POR(  0x9000) |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_NEW\_METADATA\_V3) | #R\_GET\_UPDATE\_N2  SW=0x9000 |

Test Sequence #03 Nominal: Unset PPR1 and PPR2 and update Profile name and Service Provider name

The purpose of this test is to verify that MNO can unset PPR1 and PPR2 from a Profile and can update the Service Provider Name and Profile Name values.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| 1 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [INSTALL\_PERSO\_RES\_ISDP];  MTD\_STORE\_DATA\_SCRIPT(  #UPD\_NAMES\_REM\_PPRS\_V3,  FALSE)) | SW=0x91XX |
| 2 | S\_Device → eUICC | FETCH “XX” | MTD\_CHECK\_SMS\_POR(  0x9000) |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_NEW\_METADATA\_V3) | #R\_GET\_UPDATE\_N3  SW=0x9000 |

Test Sequence #04 Nominal: VOID

Test Sequence #05 Nominal: Delete icon

The purpose of this test is to verify that the MNO can delete the icon and icon type from a Profile.

|  |  |
| --- | --- |
| **Initial Conditions** |  |
| **Entity** | **Description of the initial condition** |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| 1 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [INSTALL\_PERSO\_RES\_ISDP];  MTD\_STORE\_DATA\_SCRIPT(  #UPD\_NAMES\_REM\_ICON\_REM\_PPRS\_V3,  FALSE)) | SW=0x91XX |
| 2 | S\_Device 🡪eUICC | FETCH “XX” | MTD\_CHECK\_SMS\_POR(0x9000) |
| 3 | S\_Device 🡪 eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_NEW\_METADATA\_V3) | #R\_GET\_UPDATE\_N7  SW=0x9000 |

Test Sequence #06 Nominal: Delete Unset PPRs

The purpose of this test is to verify that the MNO can delete already unset PPRs using the Update Metadata request.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [INSTALL\_PERSO\_RES\_ISDP];  MTD\_STORE\_DATA\_SCRIPT(  #REMOVE\_ PPRS\_V3,  FALSE)) | SW=0x91XX |
| IC3 | S\_Device → eUICC | FETCH “XX” | MTD\_CHECK\_SMS\_POR(  0x9000) |
| IC4 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 1 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [INSTALL\_PERSO\_RES\_ISDP];  MTD\_STORE\_DATA\_SCRIPT(  #UPD\_NAMES\_REM\_PPRS\_V3,  FALSE)) | SW=0x91XX |
| 2 | S\_Device → eUICC | FETCH “XX” | MTD\_CHECK\_SMS\_POR(  0x9000) |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_NEW\_METADATA\_V3) | #R\_GET\_UPDATE\_N6  SW=0x9000 |

Test Sequence #07 Error: Set a pprUpdateControl value to one

The purpose of this test is to verify that the eUICC is correctly handling a pprUpdateControl value error from the MNO request, and return the expected error code status.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| 1 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [INSTALL\_PERSO\_RES\_ISDP];  MTD\_STORE\_DATA\_SCRIPT(  #UPD\_PPR\_CONTROL,  FALSE)) | SW=0x91XX |
| 2 | S\_Device → eUICC | FETCH “XX” | MTD\_CHECK\_SMS\_POR(  0x6A81) |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_NEW\_METADATA\_V3) | #R\_METADATA\_UNCHANGED  SW=0x9000 |

Test Sequence #08 Error: Update Metadata on a Disable Profile

The purpose of this test is to verify that the eUICC is correctly rejecting an Update Metadata request from the MNO when the targeted Profile is Disabled.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| 1 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [INSTALL\_PERSO\_RES\_ISDP];  MTD\_STORE\_DATA\_SCRIPT(  #REMOVE\_PPR1,  FALSE)) | SW=0x91XX  or SW=0x9000 (i.e. envelope rejected, see NOTE)  or any error SW (i.e. envelope rejected, see NOTE) |
| 2 | S\_Device → eUICC | FETCH “XX” | SMS POR received  SCP80 response status code equal to 0x06 (Unidentified security error) or 0x09 (TAR unknown) |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_NEW\_METADATA\_V3) | #R\_METADATA\_UNCHANGED  SW=0x9000 |
| NOTE: Depending on the implementation, the eUICC MAY decide to not send back a POR (i.e. SW=0x9000 on the ENVELOPE command). Therefore, the steps 2 and 3 SHALL only be executed in case SW=0x91XX. | | | |

Test Sequence #09 Error: Empty request

The purpose of this test is to verify that the eUICC is correctly rejecting an Update Metadata request from the MNO when no field is present.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| 1 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [INSTALL\_PERSO\_RES\_ISDP];  MTD\_STORE\_DATA\_SCRIPT(  #UPD\_NO\_METADATA,  FALSE)) | SW=0x91XX |
| 2 | S\_Device → eUICC | FETCH “XX” | MTD\_CHECK\_SMS\_POR(  <ANY\_SW\_IN\_ERROR>) |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_NEW\_METADATA\_V3) | #R\_METADATA\_UNCHANGED  SW=0x9000 |

Test Sequence #10 Error: VOID

Test Sequence #11 Error: VOID

Test Sequence #12 Error: Update Metadata with Enterprise Configuration while the targeted profile is not an Enterprise profile

The purpose of this test is to verify that a non-Enterprise profile cannot accept Update Metadata command UPD\_WITH\_EC with Enterprise Configuration fields.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| 1 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [INSTALL\_PERSO\_RES\_ISDP];  MTD\_STORE\_DATA\_SCRIPT(  #UPD\_WITH\_EC,  FALSE)) | SW=0x91XX |
| 2 | S\_Device → eUICC | FETCH “XX” | MTD\_CHECK\_SMS\_POR(  0x6985) |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_NEW\_METADATA\_V3) | #R\_METADATA\_UNCHANGED  SW=0x9000 |

***Test Sequence #13 Error: Delete Service Provider Name***

The purpose of this test is to verify that the eUICC is correctly handling the deletion of a non deletable Metadata from the MNO request and return the expected error code status.

|  |  |
| --- | --- |
| **Initial Conditions** |  |
| **Entity** | **Description of the initial condition** |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| 1 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [INSTALL\_PERSO\_RES\_ISDP];  MTD\_STORE\_DATA\_SCRIPT(  #REM\_SP\_NAME,  FALSE)) | SW=0x91XX |
| 2 | S\_Device 🡪eUICC | FETCH “XX” | MTD\_CHECK\_SMS\_POR(0x6A80) |
| 3 | S\_Device 🡪 eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_NEW\_METADATA\_V3) | #R\_METADATA\_UNCHANGED  SW=0x9000 |

##### 4.2.2.2.2 TC\_eUICC\_ES6.UpdateMetadata\_EnterpriseProfiles

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The communication between the S\_Device and the eUICC has been initialized by using the [TERMINAL\_CAPABILITY\_Enterprise] and the S\_LPAd has selected the ISD-R. |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_WITH\_ER is loaded on the eUICC. |

Test Sequence #01 Nominal: Enterprise Rules present in Metadata

The purpose of this test is to update Metadata of the PROFILE\_OPERATIONAL1 with new Enterprise rules as defined in UPD\_ENT\_CONFIG1.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_Enterprise | | |
| 1 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [INSTALL\_PERSO\_RES\_ISDP];  MTD\_STORE\_DATA\_SCRIPT(  #UPD\_ENT\_CONFIG1,  FALSE)) | SW=0x91XX |
| 2 | S\_Device → eUICC | FETCH “XX” | MTD\_CHECK\_SMS\_POR(  0x9000) |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_ENTERPRISE\_CONFIG\_OP\_PROF1) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   iccid #ICCID\_OP\_PROF1,  enterpriseConfiguration  #ENTERPRISE\_CONFIG3  }  }  SW=0x9000 |

Test Sequence #02 Nominal: Switch of Reference Enterprise Rules form one Enterprise profile to another

The purpose of this test is to update Metadata of the PROFILE\_OPERATIONAL1 (configured with METADATA\_WITH\_ER) with UPD\_ENT\_CONFIG1 (Reference Enterprise Rules bit is set), while PROFILE\_OPERATIONAL2 (configured with METADATA\_WITH\_RER\_PROF2) has already the Reference Enterprise Rules bit set using.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL2 with #METADATA\_WITH\_RER\_PROF2 is loaded on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_WITH\_ER is Enabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_Enterprise | | |
| 1 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [INSTALL\_PERSO\_RES\_ISDP];  MTD\_STORE\_DATA\_SCRIPT(  #UPD\_ENT\_CONFIG1,  FALSE)) | SW=0x91XX |
| 2 | S\_Device → eUICC | FETCH “XX” | MTD\_CHECK\_SMS\_POR(  0x9000) |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_ENTERPRISE\_CONFIG\_OP\_PROF1) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   iccid #ICCID\_OP\_PROF1,  enterpriseConfiguration  #ENTERPRISE\_CONFIG3  }  }  SW=0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_ENTERPRISE\_CONFIG\_OP\_PROF2) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   iccid #ICCID\_OP\_PROF2,  enterpriseConfiguration  #ENTERPRISE\_CONFIG4  }  }  SW=0x9000 |

Test Sequence #03 Nominal: Update Metadata with Reference Enterprise Rules while a non-Enterprise profile is already installed

The purpose of this test is to update Metadata of the PROFILE\_OPERATIONAL1 with Reference Enterprise Rules bit set and Only Enterprise Profiles Installed bit set, while a non-Enterprise profile PROFILE\_OPERATIONAL2 is already installed. The command shall be accepted.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL2 is loaded on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_WITH\_ER is Enabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_Enterprise | | |
| 1 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [INSTALL\_PERSO\_RES\_ISDP];  MTD\_STORE\_DATA\_SCRIPT(  #UPD\_ENT\_CONFIG1,  FALSE)) | SW=0x91XX |
| 2 | S\_Device → eUICC | FETCH “XX” | MTD\_CHECK\_SMS\_POR(  0x9000) |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_ENTERPRISE\_CONFIG\_OP\_PROF1) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   iccid #ICCID\_OP\_PROF1,  enterpriseConfiguration  #ENTERPRISE\_CONFIG3  }  }  SW=0x9000 |

Test Sequence #04 Error: Enterprise Rules present in Metadata

The purpose of this test is to verify that update of Metadata with Enterprise Rules using UPD\_ENT\_CONFIG1 while the targeted profile with METADATA\_WITH\_EC does not contain Enterprise rules, will fail.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_WITH\_EC is loaded on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_Enterprise | | |
| 1 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [INSTALL\_PERSO\_RES\_ISDP];  MTD\_STORE\_DATA\_SCRIPT(  #UPD\_ENT\_CONFIG1,  FALSE)) | SW=0x91XX |
| 2 | S\_Device → eUICC | FETCH “XX” | MTD\_CHECK\_SMS\_POR(  0x6985) |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_ENTERPRISE\_CONFIG\_OP\_PROF1) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   iccid #ICCID\_OP\_PROF1,  enterpriseConfiguration  #ENTERPRISE\_CONFIG5  }  }  SW=0x9000 |

**4.2.2.2.3 TC\_eUICC\_ES6.UpdateMetadata\_Service\_Specific\_Data**

Throughout all the ES6.UpdateMetadata test cases, SMS is used as the secure OTA channel.

|  |  |
| --- | --- |
| **General Initial Conditions** | |
| **Entity** | **Description of the general initial condition** |
| eUICC | The communication between the S\_Device and the eUICC has been initialized by using the [TERMINAL\_CAPABILITY\_LPA\_Alerting] and the S\_LPAd has selected the ISD-R. |

***Test Sequence #01 Nominal: Update Service Specific Data***

The purpose of this test is to verify that the MNO can update the Service Specific Data using the Update Metadata request.

|  |  |
| --- | --- |
| **Initial Conditions** |  |
| **Entity** | **Description of the initial condition** |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_WITH\_PPRS\_ICON\_AND\_SPEC\_DATA is loaded on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_LPA\_Alerting | | |
| 1 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [INSTALL\_PERSO\_RES\_ISDP];  MTD\_STORE\_DATA\_SCRIPT(  #UPD\_SPEC\_DATA,  FALSE)) | SW=0x91XX |
| 2 | S\_Device 🡪eUICC | FETCH “XX” | MTD\_CHECK\_SMS\_POR(  0x9000) |
| 3 | S\_Device 🡪eUICC | TERMINAL RESPONSE | SW=0x91XX |
| 4 | S\_Device 🡪eUICC | FETCH “XX” | REFRESH Command (“Application Update”, “BF22”) |
| 5 | S\_Device 🡪 eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 6 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_NEW\_METADATA\_V3) | #R\_GET\_UPDATE\_N10  SW=0x9000 |

***Test Sequence #02 Nominal: Delete Service Specific Data***

The purpose of this test is to verify that the MNO can delete the Service Specific Data Info using the Update Metadata request.

|  |  |
| --- | --- |
| **Initial Conditions** |  |
| **Entity** | **Description of the initial condition** |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_WITH\_PPRS\_ICON\_AND\_SPEC\_DATA is loaded on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_LPA\_Alerting | | |
| 1 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [INSTALL\_PERSO\_RES\_ISDP];  MTD\_STORE\_DATA\_SCRIPT(  #REM\_SPEC\_DATA,  FALSE)) | SW=0x91XX |
| 2 | | S\_Device 🡪eUICC | FETCH “XX” | MTD\_CHECK\_SMS\_POR(  0x9000) |
| 3 | | S\_Device 🡪eUICC | TERMINAL RESPONSE | SW=0x91XX |
| 4 | S\_Device 🡪eUICC | FETCH “XX” | REFRESH Command (“Application Update”, “BF22”) |
| 5 | S\_Device 🡪 eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 6 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_NEW\_METADATA\_V3) | #R\_GET\_UPDATE\_N11  SW=0x9000 |

**4.2.2.2.4 TC\_eUICC\_ES6.UpdateMetadata\_V3NotificationConfiguration**

Throughout all the ES6.UpdateMetadata test cases, SMS is used as the secure OTA channel.

|  |  |
| --- | --- |
| **General Initial Conditions** | |
| **Entity** | **Description of the general initial condition** |
| eUICC | The communication between the S\_Device and the eUICC has been initialized by using the [TERMINAL\_CAPABILITY\_LPA\_Alerting] and the S\_LPAd has selected the ISD-R. |

***Test Sequence #01 Nominal: Update Notification Configuration Info***

The purpose of this test is to verify that the MNO can update the Notification Configuration Info using the Update Metadata request.

|  |  |  |
| --- | --- | --- |
| **Initial Conditions** |  | |
| **Entity** | **Description of the initial condition** | |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_WITH\_PPRS\_ICON\_AND\_NOTIF is loaded on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_LPA\_Alerting | | |
| 1 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [INSTALL\_PERSO\_RES\_ISDP];  MTD\_STORE\_DATA\_SCRIPT(  #UPD\_NOTIF\_CONFIG\_INFO,  FALSE)) | SW=0x91XX |
| 2 | S\_Device 🡪eUICC | FETCH “XX” | MTD\_CHECK\_SMS\_POR(  0x9000) |
| 3 | S\_Device 🡪eUICC | TERMINAL RESPONSE | SW=0x91XX |
| 4 | S\_Device 🡪eUICC | FETCH “XX” | REFRESH Command (“Application Update”, “B6”) |
| 5 | S\_Device 🡪 eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 6 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_NEW\_METADATA\_V3) | #R\_GET\_UPDATE\_N8  SW=0x9000 |

***Test Sequence #02 Nominal: Delete Notification Configuration Info***

The purpose of this test is to verify that the MNO can delete elements in the Notification Configuration Info using the Update Metadata request.

|  |  |  |
| --- | --- | --- |
| **Initial Conditions** |  | |
| **Entity** | **Description of the initial condition** | |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_WITH\_PPRS\_ICON\_AND\_NOTIF is loaded on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_LPA\_Alerting | | |
| 1 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [INSTALL\_PERSO\_RES\_ISDP];  MTD\_STORE\_DATA\_SCRIPT(  #REM\_NOTIF\_CONFIG\_INFO,  FALSE)) | SW=0x91XX |
| 2 | S\_Device 🡪eUICC | FETCH “XX” | MTD\_CHECK\_SMS\_POR(  0x9000) |
| 3 | S\_Device 🡪eUICC | TERMINAL RESPONSE | SW=0x91XX |
| 4 | S\_Device 🡪eUICC | FETCH “XX” | REFRESH Command (“Application Update”, “B6”) |
| 5 | S\_Device 🡪 eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 6 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_NEW\_METADATA\_V3) | #R\_GET\_UPDATE\_N9  SW=0x9000 |

**4.2.2.2.5 TC\_eUICC\_ES6.UpdateMetadata\_V3RPM**

Throughout all the ES6.UpdateMetadata test cases, SMS is used as the secure OTA channel.

|  |  |
| --- | --- |
| **General Initial Conditions** | |
| **Entity** | **Description of the general initial condition** |
| eUICC | The communication between the S\_Device and the eUICC has been initialized by using the [TERMINAL\_CAPABILITY\_LPA\_Alerting] and the S\_LPAd has selected the ISD-R. |

***Test Sequence #01 Nominal: Update RPM Configuration***

The purpose of this test is to verify that the MNO can update the RPM Configuration using the Update Metadata request.

|  |  |  |
| --- | --- | --- |
| **Initial Conditions** |  | |
| **Entity** | **Description of the initial condition** | |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_WITH\_PPRS\_ICON\_AND\_RPM\_CONFIG is loaded on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_LPA\_Alerting | | |
| 1 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [INSTALL\_PERSO\_RES\_ISDP];  MTD\_STORE\_DATA\_SCRIPT(  #UPD\_RPM\_CONFIG,  FALSE)) | SW=0x91XX |
| 2 | S\_Device 🡪eUICC | FETCH “XX” | MTD\_CHECK\_SMS\_POR(  0x9000) |
| 3 | S\_Device 🡪eUICC | TERMINAL RESPONSE | SW=0x91XX |
| 4 | S\_Device 🡪eUICC | FETCH “XX” | REFRESH Command (“Application Update”, “BA”) |
| 5 | S\_Device 🡪 eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 6 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_NEW\_METADATA\_V3) | #R\_GET\_UPDATE\_N12  SW=0x9000 |

***Test Sequence #02 Nominal: Delete RPM Configuration***

The purpose of this test is to verify that the MNO can delete elements in the RPM Configuration using the Update Metadata request.

|  |  |  |
| --- | --- | --- |
| **Initial Conditions** |  | |
| **Entity** | **Description of the initial condition** | |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_WITH\_PPRS\_ICON\_AND\_RPM\_CONFIG is loaded on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_LPA\_Alerting | | |
| 1 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [INSTALL\_PERSO\_RES\_ISDP];  MTD\_STORE\_DATA\_SCRIPT(  #REM\_ RPM\_CONFIG,  FALSE)) | SW=0x91XX |
| 2 | S\_Device 🡪eUICC | FETCH “XX” | MTD\_CHECK\_SMS\_POR(  0x9000) |
| 3 | S\_Device 🡪eUICC | TERMINAL RESPONSE | SW=0x91XX |
| 3 | S\_Device 🡪eUICC | FETCH “XX” | REFRESH Command (“Application Update”, “BA”) |
| 4 | S\_Device 🡪 eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 5 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_NEW\_METADATA\_V3) | #R\_GET\_UPDATE\_N13  SW=0x9000 |

**4.2.2.2.6 TC\_eUICC\_ES6.UpdateMetadata\_V3HRIServerAddress**

Throughout all the ES6.UpdateMetadata test cases, SMS is used as the secure OTA channel.

|  |  |
| --- | --- |
| **General Initial Conditions** | |
| **Entity** | **Description of the general initial condition** |
| eUICC | The communication between the S\_Device and the eUICC has been initialized by using the [TERMINAL\_CAPABILITY\_LPA\_Alerting] and the S\_LPAd has selected the ISD-R. |

***Test Sequence #01 Nominal: Update HRI Server Address***

The purpose of this test is to verify that the MNO can update the HRI Server Address using the Update Metadata request.

|  |  |  |
| --- | --- | --- |
| **Initial Conditions** |  | |
| **Entity** | **Description of the initial condition** | |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_WITH\_PPRS\_ICON\_AND\_HRI\_ADDR is loaded on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_LPA\_Alerting | | |
| 1 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [INSTALL\_PERSO\_RES\_ISDP];  MTD\_STORE\_DATA\_SCRIPT(  #UPD\_HRI\_ADDR,  FALSE)) | SW=0x91XX |
| 2 | S\_Device 🡪eUICC | FETCH “XX” | MTD\_CHECK\_SMS\_POR(  0x9000) |
| 3 | S\_Device 🡪eUICC | TERMINAL RESPONSE | SW=0x91XX |
| 4 | S\_Device 🡪eUICC | FETCH “XX” | REFRESH Command (“Application Update”, “9B”) |
| 5 | S\_Device 🡪 eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 6 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_NEW\_METADATA\_V3) | #R\_GET\_UPDATE\_N14  SW=0x9000 |

***Test Sequence #02 Nominal: Delete HRI Server Address***

The purpose of this test is to verify that the MNO can delete the HRI Server Address using the Update Metadata request.

|  |  |  |
| --- | --- | --- |
| **Initial Conditions** |  | |
| **Entity** | **Description of the initial condition** | |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_WITH\_PPRS\_ICON\_AND\_HRI\_ADDR is loaded on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_LPA\_Alerting | | |
| 1 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [INSTALL\_PERSO\_RES\_ISDP];  MTD\_STORE\_DATA\_SCRIPT(  #REM\_HRI\_ADDR,  FALSE)) | SW=0x91XX |
| 2 | S\_Device 🡪eUICC | FETCH “XX” | MTD\_CHECK\_SMS\_POR(  0x9000) |
| 3 | S\_Device 🡪eUICC | TERMINAL RESPONSE | SW=0x91XX |
| 4 | S\_Device 🡪eUICC | FETCH “XX” | REFRESH Command (“Application Update”, “9B”) |
| 5 | S\_Device 🡪 eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 6 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_NEW\_METADATA\_V3) | #R\_GET\_UPDATE\_N11  SW=0x9000 |

**4.2.2.2.7 TC\_eUICC\_ES6.UpdateMetadata\_V3LPRConfiguration**

Throughout all the ES6.UpdateMetadata test cases, SMS is used as the secure OTA channel.

|  |  |
| --- | --- |
| **General Initial Conditions** | |
| **Entity** | **Description of the general initial condition** |
| eUICC | The communication between the S\_Device and the eUICC has been initialized by using the [TERMINAL\_CAPABILITY\_LPA\_Alerting] and the S\_LPAd has selected the ISD-R. |

***Test Sequence #01 Nominal: Update LPR Configuration***

The purpose of this test is to verify that the MNO can update the LPR Configuration using the Update Metadata request.

|  |  |  |
| --- | --- | --- |
| **Initial Conditions** |  | |
| **Entity** | **Description of the initial condition** | |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_WITH\_PPRS\_ICON\_AND\_LPR\_CONFIG is loaded on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_LPA\_Alerting | | |
| 1 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [INSTALL\_PERSO\_RES\_ISDP];  MTD\_STORE\_DATA\_SCRIPT(  #UPD\_LPR\_CONFIG,  FALSE)) | SW=0x91XX |
| 2 | S\_Device 🡪eUICC | FETCH “XX” | MTD\_CHECK\_SMS\_POR(  0x9000) |
| 3 | S\_Device 🡪eUICC | TERMINAL RESPONSE | SW=0x91XX |
| 4 | S\_Device 🡪eUICC | FETCH “XX” | REFRESH Command (“Application Update”, “BC”) |
| 5 | S\_Device 🡪 eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 6 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_NEW\_METADATA\_V3) | #R\_GET\_UPDATE\_N15  SW=0x9000 |

***Test Sequence #02 Nominal: Delete LPR Configuration***

The purpose of this test is to verify that the MNO can delete elements in the LPR Configuration using the Update Metadata request.

|  |  |  |
| --- | --- | --- |
| **Initial Conditions** |  | |
| **Entity** | **Description of the initial condition** | |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_WITH\_PPRS\_ICON\_AND\_LPR\_CONFIG is loaded on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_LPA\_Alerting | | |
| 1 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [INSTALL\_PERSO\_RES\_ISDP];  MTD\_STORE\_DATA\_SCRIPT(  #REM\_LPR\_CONFIG,  FALSE)) | SW=0x91XX |
| 2 | S\_Device 🡪eUICC | FETCH “XX” | MTD\_CHECK\_SMS\_POR(  0x9000) |
| 3 | S\_Device 🡪eUICC | TERMINAL RESPONSE | SW=0x91XX |
| 4 | S\_Device 🡪eUICC | FETCH “XX” | REFRESH Command (“Application Update”, “BC”) |
| 5 | S\_Device 🡪 eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 6 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_NEW\_METADATA\_V3) | #R\_GET\_UPDATE\_N16  SW=0x9000 |

**4.2.2.2.8 TC\_eUICC\_ES6.UpdateMetadata\_V3DeviceChange**

Throughout all the ES6.UpdateMetadata test cases, SMS is used as the secure OTA channel.

|  |  |
| --- | --- |
| **General Initial Conditions** | |
| **Entity** | **Description of the general initial condition** |
| eUICC | The communication between the S\_Device and the eUICC has been initialized by using the [TERMINAL\_CAPABILITY\_LPA\_Alerting] and the S\_LPAd has selected the ISD-R. |

***Test Sequence #01 Nominal: Update Device Change Configuration with DP***

The purpose of this test is to verify that the MNO can update the Device Change Configuration using the Update Metadata request.

|  |  |  |
| --- | --- | --- |
| **Initial Conditions** |  | |
| **Entity** | **Description of the initial condition** | |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_WITH\_PPRS\_ICON\_AND\_DC\_CONFIG\_DP is loaded on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_LPA\_Alerting | | |
| 1 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [INSTALL\_PERSO\_RES\_ISDP];  MTD\_STORE\_DATA\_SCRIPT(  #UPD\_DC\_CONFIG\_DP,  FALSE)) | SW=0x91XX |
| 2 | S\_Device 🡪eUICC | FETCH “XX” | MTD\_CHECK\_SMS\_POR(  0x9000) |
| 3 | S\_Device 🡪eUICC | TERMINAL RESPONSE | SW=0x91XX |
| 4 | S\_Device 🡪eUICC | FETCH “XX” | REFRESH Command (“Application Update”, “BF20”) |
| 5 | S\_Device 🡪 eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 6 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_NEW\_METADATA\_V3) | #R\_GET\_UPDATE\_N17  SW=0x9000 |

***Test Sequence #02 Nominal: Delete Device Change Configuration with DP***

The purpose of this test is to verify that the MNO can delete elements in the Device Change Configuration using the Update Metadata request.

|  |  |  |
| --- | --- | --- |
| **Initial Conditions** |  | |
| **Entity** | **Description of the initial condition** | |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_WITH\_PPRS\_ICON\_AND\_DC\_CONFIG\_DP is loaded on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_LPA\_Alerting | | |
| 1 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [INSTALL\_PERSO\_RES\_ISDP];  MTD\_STORE\_DATA\_SCRIPT(  #REM\_DC\_CONFIG\_DP,  FALSE)) | SW=0x91XX |
| 2 | S\_Device 🡪eUICC | FETCH “XX” | MTD\_CHECK\_SMS\_POR(  0x9000) |
| 3 | S\_Device 🡪eUICC | TERMINAL RESPONSE | SW=0x91XX |
| 4 | S\_Device 🡪eUICC | FETCH “XX” | REFRESH Command (“Application Update”, “BF20”) |
| 5 | S\_Device 🡪 eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 6 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_NEW\_METADATA\_V3) | #R\_GET\_UPDATE\_N18  SW=0x9000 |

***Test Sequence #03 Nominal: Update Device Change Configuration with AC***

The purpose of this test is to verify that the MNO can update the Device Change Configuration using the Update Metadata request.

|  |  |  |
| --- | --- | --- |
| **Initial Conditions** |  | |
| **Entity** | **Description of the initial condition** | |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_WITH\_PPRS\_ICON\_AND\_DC\_CONFIG\_AC is loaded on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_LPA\_Alerting | | |
| 1 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [INSTALL\_PERSO\_RES\_ISDP];  MTD\_STORE\_DATA\_SCRIPT(  #UPD\_DC\_CONFIG\_AC,  FALSE)) | SW=0x91XX |
| 2 | S\_Device 🡪eUICC | FETCH “XX” | MTD\_CHECK\_SMS\_POR(  0x9000) |
| 3 | S\_Device 🡪eUICC | TERMINAL RESPONSE | SW=0x91XX |
| 4 | S\_Device 🡪eUICC | FETCH “XX” | REFRESH Command (“Application Update”, “BF20”) |
| 5 | S\_Device 🡪 eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 6 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_NEW\_METADATA\_V3) | #R\_GET\_UPDATE\_N19  SW=0x9000 |

***Test Sequence #04 Nominal: Delete Device Change Configuration with AC***

The purpose of this test is to verify that the MNO can delete elements in the Device Change Configuration using the Update Metadata request.

|  |  |  |
| --- | --- | --- |
| **Initial Conditions** |  | |
| **Entity** | **Description of the initial condition** | |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_WITH\_PPRS\_ICON\_AND\_DC\_CONFIG\_AC is loaded on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_LPA\_Alerting | | |
| 1 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [INSTALL\_PERSO\_RES\_ISDP];  MTD\_STORE\_DATA\_SCRIPT(  #REM\_DC\_CONFIG\_AC,  FALSE)) | SW=0x91XX |
| 2 | S\_Device 🡪eUICC | FETCH “XX” | MTD\_CHECK\_SMS\_POR(  0x9000) |
| 3 | S\_Device 🡪eUICC | TERMINAL RESPONSE | SW=0x91XX |
| 4 | S\_Device 🡪eUICC | FETCH “XX” | REFRESH Command (“Application Update”, “BF20”) |
| 5 | S\_Device 🡪 eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 6 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_NEW\_METADATA\_V3) | #R\_GET\_UPDATE\_N20  SW=0x9000 |

### 4.2.3 ES8+ (SM-DP+ -- eUICC): InitialiseSecureChannel

#### 4.2.3.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 2.5.6
* Section 3.1.3.3
* Section 3.5
* Section 5.5.1

#### 4.2.3.2 Test Cases

##### 4.2.3.2.1 TC\_eUICC\_ES8+.InitialiseSecureChannel

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is not loaded on the eUICC. |
| eUICC | The communication between the S\_Device and the eUICC has been initialized and the S\_LPAd has selected the ISD-R.  Common Mutual Authentication procedure has been successfully executed between the eUICC and the S\_SM-DP+  Sub-procedure Profile Download and Installation – End User Confirmation has been successfully executed between the eUICC and the S\_SM-DP+   * #PREP\_DOWNLOAD\_NO\_CC has been sent to the eUICC |

Test Sequence #01 Error: Invalid Remote Operation

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #INIT\_SC\_INVALID\_OP\_ID,  #CONF\_ISDP\_PROF1,  #METADATA\_OP\_PROF1,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Split the <BPP> into several segments arrays named:   * <BPP\_SEG\_INIT> * <BPP\_SEG\_A0> * <BPP\_SEG\_A1> * <BPP\_SEG\_A3> | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands except the last one  SW=0x9000 with the response data #R\_PIR\_INVALID\_OP\_ID for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |

Test Sequence #02 Error: Invalid SM-DP+ Signature

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #INIT\_SC\_INVALID\_SIGN,  #CONF\_ISDP\_PROF1,  #METADATA\_OP\_PROF1,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Execute the step IC3 of the Test Sequence #01 defined in this section | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data all STORE DATA commands except for the last one  SW=0x9000 with the response data #R\_PIR\_INVALID\_SIGN for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |

Test Sequence #03 Error: Invalid Transaction Identifier

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #INIT\_SC\_INVALID\_TRANS\_ID,  #CONF\_ISDP\_PROF1,  #METADATA\_OP\_PROF1,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Execute the step IC3 of the Test Sequence #01 defined in this section | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands except the last one  SW=0x9000 with the response data #R\_PIR\_INVALID\_TRANS\_ID for the last STORE DATA command  The transactionId returned in the response SHALL not be checked (any value SHALL be accepted)  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |

Test Sequence #04 Error: Invalid CRT Values

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Step | Direction | | Sequence / Description | | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #INIT\_SC\_INVALID\_CRT,  #CONF\_ISDP\_PROF1,  #METADATA\_OP\_PROF1,  NO\_PARAM,  #UPP\_OP\_PROF1) | | | | |
| IC3 | Execute the step IC3 of the Test Sequence #01 defined in this section | | | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | | SW=0x9000 without response data for the intermediate STORE DATA commands (if any)  SW=0x9000 with the response data #R\_PIR\_INVALID\_CRT  for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG | |

Test Sequence #05 Error: InitialiseSecureChannel request while Secure Channel Session is ongoing

The purpose of this test is to ensure that the eUICC rejects an InitialiseSecureChannel request if a secure channel session is already ongoing.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1, | | |
| IC3 | Execute the step IC3 of the Test Sequence #01 defined in this section | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x6A88 or 0x6985  or  SW=0x9000 with a ProfileInstallationResult containing an ErrorResult |

### 4.2.4 ES8+ (SM-DP+ -- eUICC): ConfigureISDP

#### 4.2.4.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 2.4.4
* Section 2.5.6
* Section 3.1.3.3
* Section 3.5
* Section 5.5.2

#### 4.2.4.2 Test Cases

##### 4.2.4.2.1 TC\_eUICC\_ES8+.ConfigureISDP

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is not loaded on the eUICC. |
| eUICC | The communication between the S\_Device and the eUICC has been initialized and the S\_LPAd has selected the ISD-R.  Common Mutual Authentication procedure has been successfully executed between the eUICC and the S\_SM-DP+  Sub-procedure Profile Download and Installation – End User Confirmation has been successfully executed between the eUICC and the S\_SM-DP+   * #PREP\_DOWNLOAD\_NO\_CC has been sent to the eUICC |

Test Sequence #01 Nominal: Empty Proprietary Data

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_EMPTY,  #METADATA\_OP\_PROF1,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Split the <BPP> into several segments arrays named:   * <BPP\_SEG\_INIT> * <BPP\_SEG\_A0> * <BPP\_SEG\_A1> * <BPP\_SEG\_A3> | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | SW=0x9000 without response data for all STORE DATA commands except the last one  SW=0x9000 with the response data #R\_PIR\_OK  for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG. |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_CONF\_OP\_PROF1) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  isdpAid <ISD\_P\_AID>  *-- dpProprietaryData SHALL not be*  *-- present*  }  }  SW=0x9000 |

Test Sequence #02 Nominal: Proprietary Data with the maximum length authorized (i.e. 128 bytes)

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_MAX\_LENGTH,  #METADATA\_OP\_PROF1,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Execute the step IC3 of the Test Sequence #01 defined in this section | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | SW=0x9000 without response data for all STORE DATA commands except the last one  SW=0x9000 with the response data #R\_PIR\_OK  for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG. |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_CONF\_OP\_PROF1) | #R\_CONF\_OP\_PROF1  SW=0x9000 |

Test Sequence #03 Error: Proprietary Data with the maximum length exceeded (i.e. 129 bytes)

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_SIZE\_EXCEEDED,  #METADATA\_OP\_PROF1,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Execute the step IC3 of the Test Sequence #01 defined in this section | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands except the last one  SW=0x9000 with the response data #R\_PIR\_INVALID\_DATA for the last STORE DATA command |

### 4.2.5 ES8+ (SM-DP+ -- eUICC): StoreMetadata

#### 4.2.5.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 2.4.5.1
* Section 2.5.6
* Section 2.9.1
* Section 3.1.3.3
* Section 3.2.5
* Section 5.5.3

#### 4.2.5.2 Test Cases

##### 4.2.5.2.1 TC\_eUICC\_ES8+.StoreMetadata

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The communication between the S\_Device and the eUICC has been initialized and the S\_LPAd has selected the ISD-R.  Common Mutual Authentication procedure has been successfully executed between the eUICC and the S\_SM-DP+  Sub-procedure Profile Download and Installation – End User Confirmation has been successfully executed between the eUICC and the S\_SM-DP+   * #PREP\_DOWNLOAD\_NO\_CC has been sent to the eUICC |

Test Sequence #01 Nominal: All Metadata fields present (PNG icon used and PPR1 set)

The purpose of this test is to download the PROFILE\_OPERATIONAL1 by setting all Metadata fields. In this sequence, a PNG icon is used and PPR1 is set.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | No Operational Profile is present on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #FULL\_METADATA,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Split the <BPP> into several segments arrays named:   * <BPP\_SEG\_INIT> * <BPP\_SEG\_A0> * <BPP\_SEG\_A1> * <BPP\_SEG\_A3> | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | SW=0x9000 without response data for all STORE DATA commands expect the last one  SW=0x9000 with the response data #R\_PIR\_OK  for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_METADATA\_OP\_PROF1) | #R\_GET\_METADATA\_OP\_PROF1  SW=0x9000 |

Test Sequence #02 Nominal: With JPG icon

The purpose of this case is to verify the ability to download JPG icon. The icon size does not allow for the command to fit into one data sequence.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is not loaded on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_WITH\_JPG,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Execute the step IC3 of the Test Sequence #01 defined in this section | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | SW=0x9000 without response data for all STORE DATA commands except the last one  SW=0x9000 with the response data #R\_PIR\_OK  for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  …  iccid #ICCID\_OP\_PROF1,  iconType jpg,  icon #ICON\_JPG,  …  }  }  SW=0x9000 |

Test Sequence #03 Nominal: Without providing Profile Class

The purpose of this test is to download the PROFILE\_OPERATIONAL1 by not indicating the Profile Class in the Metadata. In such a case, the default Profile Class 'Operational' SHALL be set by the eUICC.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is not loaded on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_NO\_CLASS,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Execute the step IC3 of the Test Sequence #01 defined in this section | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data #R\_PIR\_OK  for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  …  iccid #ICCID\_OP\_PROF1,  profileClass operational  …  }  }  SW=0x9000 |

Test Sequence #04 Nominal: With PPR2 set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is not loaded on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_WITH\_PPR2,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Execute the step IC3 of the Test Sequence #01 defined in this section | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data #R\_PIR\_OK  for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PPR\_OP\_PROF1) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   iccid #ICCID\_OP\_PROF1,  profilePolicyRules {ppr2}  }  }  SW=0x9000 |

Test Sequence #05 Nominal: With PPR1 and PPR2 set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | No Operational Profile is present on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_WITH\_PPR1\_PPR2,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Execute the step IC3 of the Test Sequence #01 defined in this section | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data #R\_PIR\_OK  for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PPR\_OP\_PROF1) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  iccid #ICCID\_OP\_PROF1,  profilePolicyRules {ppr1,ppr2}  }  }  SW=0x9000 |

Test Sequence #06 Nominal: With several Notification events configured

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is not loaded on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_WITH\_NOTIFS,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Execute the step IC3 of the Test Sequence #01 defined in this section | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data #R\_PIR\_OK  for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_NOTIF\_CONF\_OP\_PROF1) | #R\_GET\_PROF\_NOTIF\_CONF  SW=0x9000 |

Test Sequence #07 Error: ICCID already present in the eUICC

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | General Initial Conditions do not apply. |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | The communication between the S\_Device and the eUICC has been initialized and the S\_LPAd has selected the ISD-R.  Common Mutual Authentication procedure has been successfully executed between the eUICC and the S\_SM-DP+  Sub-procedure Profile Download and Installation – End User Confirmation has been successfully executed between the eUICC and the S\_SM-DP+  #PREP\_DOWNLOAD\_NO\_CC has been sent to the eUICC | | |
| IC2 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC3 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_OP\_PROF1,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC4 | Execute the step IC3 of the Test Sequence #01 defined in this section | | |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC6 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data #R\_PIR\_ICCID\_ALREADY\_EXIST  for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |

Test Sequence #08 Error: Profile Policy Rule is set but Profile Owner is not

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is not loaded on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_PPR\_NO\_OWNER,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Execute the step IC3 of the Test Sequence #01 defined in this section | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data #R\_PIR\_METADATA\_INVALID (See NOTE)  for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |
| NOTE: The errorReason "pprNotAllowed" or "installFailedDueToUnknownError" MAY be also returned by the eUICC. | | | |

Test Sequence #09 Error: Profile Owner is set with a wildcard ('E') digits

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is not loaded on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_WILDCARD,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Execute the step IC3 of the Test Sequence #01 defined in this section | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data #R\_PIR\_METADATA\_INVALID (See Note)  for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |
| NOTE: The errorReason "pprNotAllowed" MAY be also returned by the eUICC. | | | |

Test Sequence #10 Error: Icon Type is set but icon is not

The purpose of this test is to verify ASN.1 conditional requirement for icon presence. If icon type is present then icon SHALL also be present.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is not loaded on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_WITHOUT\_ICON,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Execute the step IC3 of the Test Sequence #01 defined in this section | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data #R\_PIR\_METADATA\_INVALID  for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |

Test Sequence #11 Error: Store Metadata with Enterprise Configuration while the targeted eUICC is non Enterprise Capable

The purpose of this test is check that a non-Enterprise capable eUICC cannot accept profile with Enterprise Configuration.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_WITH\_EC is not loaded on the eUICC. |
| S\_LPA | The S\_LPA does not perform the Cancel Session procedure. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_WITH\_EC,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Execute the step IC3 of the Test Sequence #01 defined in this section | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data #R\_PIR\_UNKNOWN\_TLV  for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |

##### 4.2.5.2.2 TC\_eUICC\_ES8+.StoreMetadata\_Service\_Specific\_Data

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The communication between the S\_Device and the eUICC has been initialized and the S\_LPAd has selected the ISD-R.  Common Mutual Authentication procedure has been successfully executed between the eUICC and the S\_SM-DP+  Sub-procedure Profile Download and Installation – End User Confirmation has been successfully executed between the eUICC and the S\_SM-DP+  #PREP\_DOWNLOAD\_NO\_CC has been sent to the eUICC |

***Test Sequence #01 Nominal: Metadata include service-specific data, stored***

The purpose of this test is to download the PROFILE\_OPERATIONAL1 with service-specific metadata stored in the eUICC.

|  |  |
| --- | --- |
| **Initial Conditions** |  |
| **Entity** | **Description of the initial condition** |
| eUICC | No Operational Profile is present on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_SERVICE\_SPECIFIC\_STORED,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Split the <BPP> into several segments arrays named:   <BPP\_SEG\_INIT>   <BPP\_SEG\_A0>   <BPP\_SEG\_A1>   <BPP\_SEG\_A3> | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | SW=0x9000 without response data for all STORE DATA commands expect the last one  SW=0x9000 with the response data #R\_PIR\_OK  for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_ECDSA |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_METADATA\_OP\_PROF1\_SERVICE\_SPECIFIC) | #R\_GET\_METADATA\_OP\_PROF1\_SERVICE\_SPECIFIC  SW=0x9000 |

***Test Sequence #02 Nominal: Metadata include service-specific data, not stored***

The purpose of this test is to download the PROFILE\_OPERATIONAL1 with service-specific metadata *not* stored in the eUICC.

|  |  |
| --- | --- |
| **Initial Conditions** |  |
| **Entity** | **Description of the initial condition** |
| eUICC | No Operational Profile is present on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_SERVICE\_SPECIFIC\_NOT\_STORED,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Split the <BPP> into several segments arrays named:   <BPP\_SEG\_INIT>   <BPP\_SEG\_A0>   <BPP\_SEG\_A1>   *  <BPP\_SEG\_A3> | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | SW=0x9000 without response data for all STORE DATA commands expect the last one  SW=0x9000 with the response data #R\_PIR\_OK  for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_ECDSA |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_METADATA\_OP\_PROF1\_SERVICE\_SPECIFIC) | #R\_GET\_METADATA\_OP\_PROF1\_NO\_SERVICE\_SPECIFIC  SW=0x9000 |

***Test Sequence #03 Nominal: Metadata include service-specific data, stored and not stored***

The purpose of this test is to download the PROFILE\_OPERATIONAL1 with service-specific metadata stored in the eUICC and other service-specific metadata *not* stored.

|  |  |
| --- | --- |
| **Initial Conditions** |  |
| **Entity** | **Description of the initial condition** |
| eUICC | No Operational Profile is present on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_SERVICE\_SPECIFIC\_STORED\_AND\_NOT\_STORED,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Split the <BPP> into several segments arrays named:   <BPP\_SEG\_INIT>   <BPP\_SEG\_A0>   <BPP\_SEG\_A1>   <BPP\_SEG\_A3> | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | SW=0x9000 without response data for all STORE DATA commands expect the last one  SW=0x9000 with the response data #R\_PIR\_OK  for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_ECDSA |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_METADATA\_OP\_PROF1\_SERVICE\_SPECIFIC) | #R\_GET\_METADATA\_OP\_PROF1\_SERVICE\_SPECIFIC  SW=0x9000 |

##### 4.2.5.2.3 TC\_eUICC\_ES8+.StoreMetadata\_EnterpriseProfiles

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The communication between the S\_Device and the eUICC has been initialized by using the [TERMINAL\_CAPABILITY\_Enterprise] and the S\_LPAd has selected the ISD-R.  Common Mutual Authentication procedure has been successfully executed between the eUICC and the S\_SM-DP+ with enterpriseCapableDevice present in DeviceInfo (in CtxParams1)  Sub-procedure Profile Download and Installation – End User Confirmation has been successfully executed between the eUICC and the S\_SM-DP+   * #PREP\_DOWNLOAD\_NO\_CC has been sent to the eUICC |

Test Sequence #01 Nominal: Enterprise Rules present in Metadata

The purpose of this test is to download the PROFILE\_OPERATIONAL1 with Enterprise rules in Metadata #METADATA\_WITH\_ER

|  |  |
| --- | --- |
| **Initial Conditions** |  |
| **Entity** | **Description of the initial condition** |
| eUICC | No Operational Profile is present on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_WITH\_ER,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Split the <BPP> into several segments arrays named:   * <BPP\_SEG\_INIT> * <BPP\_SEG\_A0> * <BPP\_SEG\_A1> * <BPP\_SEG\_A3> | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | SW=0x9000 without response data for all STORE DATA commands expect the last one  SW=0x9000 with the response data #R\_PIR\_OK  for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_ENTERPRISE\_CONFIG\_OP\_PROF1) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   iccid #ICCID\_OP\_PROF1,  enterpriseConfiguration  #ENTERPRISE\_CONFIG1\_ER  }  }  SW=0x9000 |

Test Sequence #02 Nominal: Enterprise Profile installation while another profile with no PPR1 set is installed

The purpose of this test is to download the PROFILE\_OPERATIONAL2 with Enterprise configuration in Metadata #METADATA\_WITH\_EC\_PROF2, while a non-enterprise profile PROFILE\_OPERATIONAL1 without PPR1 set is installed.

|  |  |
| --- | --- |
| **Initial Conditions** |  |
| **Entity** | **Description of the initial condition** |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_OP\_PROF1 is installed on the eUICC |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF2,  #CONF\_ISDP\_PROF2,  #METADATA\_WITH\_EC\_PROF2,  NO\_PARAM,  #UPP\_OP\_PROF2) | | |
| IC3 | Split the <BPP> into several segments arrays named:   * <BPP\_SEG\_INIT> * <BPP\_SEG\_A0> * <BPP\_SEG\_A1> * <BPP\_SEG\_A3> | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | SW=0x9000 without response data for all STORE DATA commands expect the last one  SW=0x9000 with the response data #R\_PIR\_OK  for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_ENTERPRISE\_CONFIG\_OP\_PROF2) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   iccid #ICCID\_OP\_PROF2,  enterpriseConfiguration  #ENTERPRISE\_CONFIG1\_EC  }  }  SW=0x9000 |

Test Sequence #03 Error: Enterprise Rules with Reference Enterprise Rule set

The purpose of this test is to verify that it is not possible to load the PROFILE\_OPERATIONAL1 with Reference Enterprise Rule set in Metadata #METADATA\_WITH\_RER.

|  |  |
| --- | --- |
| **Initial Conditions** |  |
| **Entity** | **Description of the initial condition** |
| eUICC | No Operational Profile is present on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_WITH\_RER,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Split the <BPP> into several segments arrays named:   * <BPP\_SEG\_INIT> * <BPP\_SEG\_A0> * <BPP\_SEG\_A1> * <BPP\_SEG\_A3> | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data #R\_PIR\_RER\_NOTALLOWED  for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |

Test Sequence #04 Error: Enterprise Profile installation while another profile with PPR1 set is installed

The purpose is to verify that it is not possible to verify that the installation of PROFILE\_OPERATIONAL2 with Enterprise configuration in Metadata #METADATA\_WITH\_EC\_PROF2, while a non-enterprise profile PROFILE\_OPERATIONAL1 with PPR1 set is installed.

|  |  |
| --- | --- |
| **Initial Conditions** |  |
| **Entity** | **Description of the initial condition** |
| eUICC | The PROFILE\_OPERATIONAL1 with #FULL\_METADATA is installed on the eUICC |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF2,  #CONF\_ISDP\_PROF2,  #METADATA\_WITH\_EC\_PROF2,  NO\_PARAM,  #UPP\_OP\_PROF2) | | |
| IC3 | Split the <BPP> into several segments arrays named:   * <BPP\_SEG\_INIT> * <BPP\_SEG\_A0> * <BPP\_SEG\_A1> * <BPP\_SEG\_A3> | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data #R\_PIR\_EP\_NOTALLOWED  for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |

Test Sequence #05 Error: non-Enterprise Profile installation while an Enterprise profile with Enterprise rules that does not allow it is installed

The purpose is to verify that it is not possible to verify that the installation of non-enterprise profile PROFILE\_OPERATIONAL2 with Metadata #METADATA\_OP\_PROF2, while an enterprise profile PROFILE\_OPERATIONAL1 is installed with Enterprise rules that does not allow it.

|  |  |
| --- | --- |
| **Initial Conditions** |  |
| **Entity** | **Description of the initial condition** |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_WITH\_RER is installed on the eUICC |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF2,  #CONF\_ISDP\_PROF2,  #METADATA\_OP\_PROF2,  NO\_PARAM,  #UPP\_OP\_PROF2) | | |
| IC3 | Split the <BPP> into several segments arrays named:   * <BPP\_SEG\_INIT> * <BPP\_SEG\_A0> * <BPP\_SEG\_A1> * <BPP\_SEG\_A3> | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data #R\_PIR\_EP\_ONLY  for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |

Test Sequence #06 Nominal: Store Metadata with Enterprise Configuration while the targeted device is a Non Enterprise Capable device

The purpose of this test is to verify that non-Enterprise capable devices can accept profile with Enterprise Configuration without Enterprise Rules and with Metadata #METADATA\_WITH\_EC.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is not loaded on the eUICC. |
| eUICC | The communication between the S\_Device and the eUICC has been initialized and the S\_LPAd has selected the ISD-R.  Common Mutual Authentication procedure has been successfully executed between the eUICC and the S\_SM-DP+ with  Sub-procedure Profile Download and Installation – End User Confirmation has been successfully executed between the eUICC and the S\_SM-DP+  #PREP\_DOWNLOAD\_NO\_CC has been sent to the eUICC |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_WITH\_EC,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Split the <BPP> into several segments arrays named:   * <BPP\_SEG\_INIT> * <BPP\_SEG\_A0> * <BPP\_SEG\_A1>   <BPP\_SEG\_A3> | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | SW=0x9000 without response data for all STORE DATA commands expect the last one  SW=0x9000 with the response data #R\_PIR\_OK  for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_ENTERPRISE\_CONFIG\_OP\_PROF1) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   iccid #ICCID\_OP\_PROF1,  enterpriseConfiguration  #ENTERPRISE\_CONFIG1\_EC  }  }  SW=0x9000 |

Test Sequence #07 Error: Store Metadata with Enterprise Rules while the targeted device is a Non Enterprise Capable device

The purpose of this test is to verify that only Enterprise capable devices can accept profile with Enterprise Rules, and with Metadata #METADATA\_WITH\_ER.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is not loaded on the eUICC. |
| eUICC | The communication between the S\_Device and the eUICC has been initialized and the S\_LPAd has selected the ISD-R.  Common Mutual Authentication procedure has been successfully executed between the eUICC and the S\_SM-DP+ with  Sub-procedure Profile Download and Installation – End User Confirmation has been successfully executed between the eUICC and the S\_SM-DP+  #PREP\_DOWNLOAD\_NO\_CC has been sent to the eUICC |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_WITH\_ER,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Split the <BPP> into several segments arrays named:   * <BPP\_SEG\_INIT> * <BPP\_SEG\_A0> * <BPP\_SEG\_A1>   <BPP\_SEG\_A3> | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data #R\_PIR\_ER\_NOTALLOWED  for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |

Test Sequence #08 Error: Store Metadata with different Enterprise OID

The purpose of this test is to verify that the installation of an Enterprise profile, with an Enterprise OID in METADATA\_WITH\_EC\_OID2 different from the one defined for an already installed profile (METADATA\_WITH\_EC), will fail.

|  |  |
| --- | --- |
| **Initial Conditions** |  |
| **Entity** | **Description of the initial condition** |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_WITH\_EC is installed on the eUICC |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF2,  #CONF\_ISDP\_PROF2,  #METADATA\_WITH\_EC\_OID2,  NO\_PARAM,  #UPP\_OP\_PROF2) | | |
| IC3 | Split the <BPP> into several segments arrays named:   * <BPP\_SEG\_INIT> * <BPP\_SEG\_A0> * <BPP\_SEG\_A1> * <BPP\_SEG\_A3> | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data #R\_PIR\_EOID\_MISMATCH  for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |

### 4.2.6 ES8+ (SM-DP+ -- eUICC): ReplaceSessionKeys

#### 4.2.6.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 2.5.6
* Section 2.6.4, 2.6.5
* Section 3.1.3.3
* Section 5.5.4

#### 4.2.6.2 Test Cases

##### 4.2.6.2.1 TC\_eUICC\_ES8+.ReplaceSessionKeys

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is not loaded on the eUICC |
| eUICC | The communication between the S\_Device and the eUICC has been initialized and the S\_LPAd has selected the ISD-R.  Common Mutual Authentication procedure has been successfully executed between the eUICC and the S\_SM-DP+  Sub-procedure Profile Download and Installation – End User Confirmation has been successfully executed between the eUICC and the S\_SM-DP+   * #PREP\_DOWNLOAD\_NO\_CC has been sent to the eUICC |

Test Sequence #01 Error: Incorrect PPK size

The purpose of this test is to verify that the eUICC checks that PPK sizes are the same as session keys.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is not loaded on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_OP\_PROF1,  #REPLACE\_S\_KEYS\_REQ\_INV\_SIZE,  #UPP\_OP\_PROF1)  MTD\_GENERATE\_BPP overriding:  For this test sequence, the initial session keys SHALL be used for UPP SCP03t protection. Therefore:  Encrypt all <UPP\_SEG> with <S\_ENC>  Calculate and add a MAC to all tags 0x86 of sequenceOf86 by using <S\_MAC> | | |
| IC3 | Split the <BPP> into several segments arrays named:   * <BPP\_SEG\_INIT> * <BPP\_SEG\_A0> * <BPP\_SEG\_A1> * <BPP\_SEG\_A2> * <BPP\_SEG\_A3> | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| IC6 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A2>) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data #R\_PIR\_PPK\_INV  for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |

### 4.2.7 ES8+ (SM-DP+ -- eUICC): LoadProfileElements

#### 4.2.7.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 2.5.6
* Section 3.1.3.3
* Section 5.5.5

#### 4.2.7.2 Test Cases

##### 4.2.7.2.1 TC\_eUICC\_ES8+.LoadProfileElements

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is not loaded on the eUICC |
| eUICC | The communication between the S\_Device and the eUICC has been initialized and the S\_LPAd has selected the ISD-R.  Common Mutual Authentication procedure has been successfully executed between the eUICC and the S\_SM-DP+  Sub-procedure Profile Download and Installation – End User Confirmation has been successfully executed between the eUICC and the S\_SM-DP+   * #PREP\_DOWNLOAD\_NO\_CC has been sent to the eUICC |

Test Sequence #01 Error: EFICCID different from the ICCID provided in the Profile Metadata

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL2 is not loaded on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_ICCID\_MISMATCH,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Split the <BPP> into several segments arrays named:   * <BPP\_SEG\_INIT> * <BPP\_SEG\_A0> * <BPP\_SEG\_A1> * <BPP\_SEG\_A3> | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| IC6 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | SW=0x9000 with the response data #R\_PIR\_DATA\_MISMATCH  for one of the STORE DATA commands  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | resp ProfileInfoListResponse ::=  profileInfoListOk :{}  SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF2,  NO\_PARAM)) | resp ProfileInfoListResponse ::=  profileInfoListOk :{}  SW=0x9000 |

Test Sequence #02 Error: MCC / MNC of EFIMSI different from MCC / MNC of Profile Owner present in Metadata

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_MCCMNC\_MISMATCH,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Execute the step IC3 of the Test Sequence #01 defined in this section | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| IC6 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | SW=0x9000 with the response data #R\_PIR\_DATA\_MISMATCH  for one of the STORE DATA commands  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | resp ProfileInfoListResponse ::=  profileInfoListOk :{}  SW=0x9000 |

Test Sequence #03 Error: Session MAC chaining used instead of new Initial MAC chaining

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP (  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_OP\_PROF1,  #REPLACE\_S\_KEYS\_REQ,  #UPP\_OP\_PROF1)  MTD\_GENERATE\_BPP overriding:  For this test sequence, <S\_MAC\_CHAIN> SHALL be used instead of <PPK\_INIT\_MAC> for UPP SCP03t protection. | | |
| IC3 | Split the <BPP> into several segments arrays named:   <BPP\_SEG\_INIT>   <BPP\_SEG\_A0>   <BPP\_SEG\_A1>   <BPP\_SEG\_A2>   <BPP\_SEG\_A3> | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| IC6 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |
| IC7 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A2>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | SW=0x9000 with the response data #R\_PIR\_SECU\_INVALID for one of the STORE DATA commands  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | resp ProfileInfoListResponse ::=  profileInfoListOk :{}  SW=0x9000 |

Test Sequence #04 Error: S-MAC used instead of PPK-MAC

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP (  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_OP\_PROF1,  #REPLACE\_S\_KEYS\_REQ,  #UPP\_OP\_PROF1)  MTD\_GENERATE\_BPP overriding:  For this test sequence <S\_MAC> SHALL be used instead of <PPK\_MAC> for UPP SCP03t protection. | | |
| IC3 | Execute the step IC3 of the Test Sequence #03 defined in this section | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| IC6 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |
| IC7 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A2>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | SW=0x9000 with the response data #R\_PIR\_SECU\_INVALID for one of the STORE DATA commands  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | resp ProfileInfoListResponse ::=  profileInfoListOk :{}  SW=0x9000 |

Test Sequence #05 Error: S-ENC used instead of PPK-ENC

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP (  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_OP\_PROF1,  #REPLACE\_S\_KEYS\_REQ,  #UPP\_OP\_PROF1)  MTD\_GENERATE\_BPP overriding:  For this test sequence <S\_ENC> SHALL be used instead of <PPK\_ENC> for UPP SCP03t protection. | | |
| IC3 | Execute the step IC3 of the Test Sequence #03 defined in this section | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| IC6 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |
| IC7 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A2>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | SW=0x9000 with the response data #R\_PIR\_SECU\_INVALID for one of the STORE DATA commands  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | resp ProfileInfoListResponse ::=  profileInfoListOk :{}  SW=0x9000 |

Test Sequence #06 Error: Profile Downloading stopped by a Reset

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | No pending Notification is present on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP (  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_OP\_PROF1,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Execute the step IC3 of the Test Sequence #01 defined in this section | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| IC6 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | SW=0x9000 without response data for all STORE DATA commands except the last one.  Step 2 SHALL be triggered before sending the last STORE DATA |
| 2 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| 3 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | resp ProfileInfoListResponse ::=  profileInfoListOk :{}  SW=0x9000 |

Test Sequence #07 Nominal: ICCID in the 'ProfileHeader' PE is ignored by the eUICC

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_OP\_PROF1  NO\_PARAM,  #UPP\_OP\_PROF1)  #UPP\_OP\_PROF1 overriding:  For this sequence, the *iccid* field SHALL be set to #ICCID\_OP\_PROF2 in the *ProfileHeader* element | | |
| IC3 | Split the <BPP> into several segments arrays named:   * <BPP\_SEG\_INIT> * <BPP\_SEG\_A0> * <BPP\_SEG\_A1> * <BPP\_SEG\_A3> | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| IC6 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data #R\_PIR\_OK for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG. <ISD\_P\_AID> SHALL be in the range as defined SGP.02 [1]. |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  …  iccid #ICCID\_OP\_PROF1,  isdpAid <ISD\_P\_AID>,  profileState disabled,  …  }  }  SW=0x9000 |

Test Sequence #08 Nominal: With gid1 and gid2 set

The purpose of this test is to verify that an Operational Profile configured with gid1 and gid2 can be downloaded on the eUICC.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL9 is not loaded on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_OP\_PROF9,  NO\_PARAM,  #UPP\_OP\_PROF9) | | |
| IC3 | Split the <BPP> into several segments arrays named:   * <BPP\_SEG\_INIT> * <BPP\_SEG\_A0> * <BPP\_SEG\_A1> * <BPP\_SEG\_A3> | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| IC6 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data #R\_PIR\_OK\_PROF9 for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_OWNERS) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  profileOwner {  mccMnc #MCC\_MNC9,  gid1 #GID1,  gid2 #GID2  }  }  }  SW=0x9000 |

Test Sequence #09 Error: gid1 and gid2 provided in the Profile Metadata but not in the Profile Package

The purpose of this test is to verify that if gid1 and gid2 are provided in the Profile Metadata but ef-gid1 and ef-gid2 are not present and the related services (17 and 18) in ef-ust are not available, the eUICC returns an error during the LoadProfileElements process.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is not loaded on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_OP1\_GID1GID2\_PRESENT,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Split the <BPP> into several segments arrays named:   * <BPP\_SEG\_INIT> * <BPP\_SEG\_A0> * <BPP\_SEG\_A1> * <BPP\_SEG\_A3> | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| IC6 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | SW=0x9000 with the response data #R\_PIR\_DATA\_MISMATCH for one of the STORE DATA commands  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | resp ProfileInfoListResponse ::=  profileInfoListOk :{}  SW=0x9000 |

Test Sequence #10 Error: gid1 and gid2 not provided in the Profile Metadata but Service is available in ef-ust

The purpose of this test is to verify that if gid1 and gid2 are not provided in the Profile Metadata but ef-gid1 and ef-gid2 are present and the related services (17 and 18) in ef-ust are available, the eUICC returns an error during the LoadProfileElements process.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL9 is not loaded on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_OP9\_GID1GID2\_MISSING,  NO\_PARAM,  #UPP\_OP\_PROF9) | | |
| IC3 | Split the <BPP> into several segments arrays named:   * <BPP\_SEG\_INIT> * <BPP\_SEG\_A0> * <BPP\_SEG\_A1> * <BPP\_SEG\_A3> | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| IC6 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | SW=0x9000 with the response data #R\_PIR\_DATA\_MISMATCH for one of the STORE DATA commands  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF9,  NO\_PARAM)) | resp ProfileInfoListResponse ::=  profileInfoListOk :{}  SW=0x9000 |

Test Sequence #11 Nominal: Profile download is success if profileMetaData does not contains profileOwner and EFIMSI file is not present

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL10 is not loaded on the eUICC. |
| eUICC | The communication between the S\_Device and the eUICC has been initialized and the S\_LPAd has selected the ISD-R.  Common Mutual Authentication procedure has been successfully executed between the eUICC and the S\_SM-DP+   * #GET\_EUICC\_INFO1, #GET\_EUICC\_CHALLENGE and #AUTHENTICATE\_SMDP\_WITH\_DEVICE\_INFO\_NAI have been sent to the eUICC * the same GSMA CI has been chosen for signing and for verification   Sub-procedure Profile Download and Installation – End User Confirmation has been successfully executed between the eUICC and the S\_SM-DP+  #PREP\_DOWNLOAD\_NO\_CC has been sent to the eUICC |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result | REQ |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_OP\_PROF10\_NO\_PROFILE\_OWNER  NO\_PARAM,  #UPP\_OP\_PROF10) | | | |
| IC3 | Split the <BPP> into several segments arrays named:   <BPP\_SEG\_INIT>   <BPP\_SEG\_A0>   <BPP\_SEG\_A1>   <BPP\_SEG\_A3> | | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |  |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |  |
| IC6 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |  |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data #R\_PIR\_OK\_PROF10 for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_ECDSA. <ISD\_P\_AID> SHALL be in the range as defined SGP.02 [1]. | RQ25\_023 RQ25\_024 RQ55\_045 RQ55\_048 RQ25\_025 RQ25\_026 RQ55\_044 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF10,  NO\_PARAM)) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   …  iccid #ICCID\_OP\_PROF10,  isdpAid <ISD\_P\_AID>,  profileState disabled,  …  }  }  SW=0x9000 | RQ32\_071 RQ55\_048 |

Test Sequence #12 Error: Profile download is failed if ProfileMetaData contains profileOwner but EFIMSI file is not present

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL10 is not loaded on the eUICC. |
| eUICC | The communication between the S\_Device and the eUICC has been initialized and the S\_LPAd has selected the ISD-R.  Common Mutual Authentication procedure has been successfully executed between the eUICC and the S\_SM-DP+   * #GET\_EUICC\_INFO1, #GET\_EUICC\_CHALLENGE and #AUTHENTICATE\_SMDP\_WITH\_DEVICE\_INFO\_NAI have been sent to the eUICC * the same GSMA CI has been chosen for signing and for verification   Sub-procedure Profile Download and Installation – End User Confirmation has been successfully executed between the eUICC and the S\_SM-DP+  #PREP\_DOWNLOAD\_NO\_CC has been sent to the eUICC |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result | REQ |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_OP\_PROF10,  NO\_PARAM,  #UPP\_OP\_PROF 10) | | | |
| IC3 | Split the <BPP> into several segments arrays named:   <BPP\_SEG\_INIT>   <BPP\_SEG\_A0>   <BPP\_SEG\_A1>  <BPP\_SEG\_A3> | | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |  |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |  |
| IC6 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |  |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | SW=0x9000 with the response data #R\_PIR\_DATA\_MISMATCH  for one of the STORE DATA commands  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_ECDSA | RQ25\_023 RQ25\_024 RQ55\_043 RQ55\_047 RQ55\_048 RQ25\_025 RQ25\_026 RQ31\_173 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF10,  NO\_PARAM)) | resp ProfileInfoListResponse ::=  profileInfoListOk :{}  SW=0x9000 | RQ32\_071 RQ55\_043 RQ55\_048 |

### 4.2.8 ES10a (LPA -- eUICC): GetEuiccConfiguredData

#### 4.2.8.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 3.1.3
* Section 3.3.4
* Section 5.7.3

#### 4.2.8.2 Test Cases

##### 4.2.8.2.1 TC\_eUICC\_ES10a.GetEuiccConfiguredAddresses

Test Sequence #01 Nominal: Only Root SM-DS Address

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | Only the Root SM-DS address has been set on the ISD-R (see Annex G.2.1). |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_EUICC\_CONFIGURED\_DATA) | #R\_ES10a\_GECA\_DS SW = 0x9000 |

Test Sequence #02 Nominal: Root SM-DS and Default SM-DP+ Addresses

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The ISD-R is provisioned with the Default SM-DP+ Address #TEST\_DP\_ADDRESS1 and the Root SM-DS address (see Annex G.2.1). |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_EUICC\_CONFIGURED\_DATA) | #R\_ES10a\_GECA\_DS\_DP\_1 SW = 0x9000 |

### 4.2.9 ES10a (LPA -- eUICC): SetDefaultDpAddress

#### 4.2.9.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 3.3.4
* Section 5.7.4

#### 4.2.9.2 Test Cases

##### 4.2.9.2.1 TC\_eUICC\_ES10a.SetDefaultDpAddress

Test Sequence #01 Nominal: Set SM-DP+ Address with Address Empty in eUICC

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | No value is assigned to the Default SM-DP+ Address field. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #SET\_EUICC\_CONFIGURED\_ADDRESS\_1) | #R\_ES10a\_SD\_DP\_A\_OK SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_EUICC\_CONFIGURED\_DATA) | #R\_ES10a\_GECA\_DS\_DP\_1 SW = 0x9000 |

Test Sequence #02 Nominal: Set SM-DP+ Address with SM-DP+ Address already in eUICC

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The SM-DP+ address #TEST\_DP\_ADDRESS1 is provisioned. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #SET\_EUICC\_CONFIGURED\_ADDRESS\_2) | #R\_ES10a\_SD\_DP\_A\_OK SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_EUICC\_CONFIGURED\_DATA) | #R\_ES10a\_GECA\_DS\_DP\_2 SW = 0x9000 |

Test Sequence #03 Nominal: Set Empty SM-DP+ Address with SM-DP+ Address already in eUICC

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The SM-DP+ address #TEST\_DP\_ADDRESS1 is provisioned. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #SET\_EUICC\_CONFIGURED\_ADDRESS\_EMPTY) | #R\_ES10a\_SD\_DP\_A\_OK SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_EUICC\_CONFIGURED\_DATA) | #R\_ES10a\_GECA\_DS SW = 0x9000 |

Test Sequence #04 Nominal: Set Empty SM-DP+ Address with Empty SM-DP+ Address in eUICC

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | No value is assigned to the Default SM-DP+ Address field. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #SET\_EUICC\_CONFIGURED\_ADDRESS\_EMPTY) | #R\_ES10a\_SD\_DP\_A\_OK SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_EUICC\_CONFIGURED\_DATA) | #R\_ES10a\_GECA\_DS  SW = 0x9000 |

### 4.2.10 ES10b (LPA -- eUICC): PrepareDownload

#### 4.2.10.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 2.6.2, 2.6.7.1, 2.6.7.2
* Section 3.0.1, 3.1.3.2
* Section 4.5.2.1, 4.5.2.2
* Section 5.7.5

#### 4.2.10.2 Test Cases

##### 4.2.10.2.1 TC\_eUICC\_ES10b.PrepareDownloadNIST

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is not loaded on the eUICC. |
| eUICC | The communication between the S\_Device and the eUICC has been initialized and the S\_LPAd has selected the ISD-R.  Common Mutual Authentication procedure has been successfully executed between the eUICC and the S\_SM-DP+   * the same GSMA CI based on NIST P-256 curve has been chosen for signing and for verification |

Test Sequence #01 Nominal: Without Confirmation Code

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  #PREP\_DOWNLOAD\_NO\_CC) | #R\_PREP\_DOWNLOAD\_NO\_CC  SW=0x9000  The <EUICC\_SIGNATURE2> SHALL be verified with the #PK\_EUICC\_SIG.  Verify that the <S\_TRANSACTION\_ID> present in the euiccSigned2 is the same as in #PREP\_DOWNLOAD\_NO\_CC. |

Test Sequence #02 Nominal: With Confirmation Code

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | <S\_HASHED\_CC> = MTD\_GENERATE\_HASHED\_CC(#CONFIRMATION\_CODE1, <S\_TRANSACTION\_ID>) | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  #PREP\_DOWNLOAD\_WITH\_CC) | #R\_PREP\_DOWNLOAD\_WITH\_CC  SW=0x9000  The <EUICC\_SIGNATURE2> SHALL be verified with the #PK\_EUICC\_SIG.  Verify that the <S\_TRANSACTION\_ID> present in the euiccSigned2 is the same as in #PREP\_DOWNLOAD\_WITH\_CC.  Verify that the <S\_HASHED\_CC> present in the euiccSigned2 is the same as in #PREP\_DOWNLOAD\_WITH\_CC. |

Test Sequence #03 Nominal: With an unknown otPK.EUICC.ECKA

The purpose of this test is to verify that the eUICC does not use the one-time key pair given by the SM-DP+ when its value does not correspond to a stored one-time key pair. In this case, the eUICC SHALL generate a new set of key.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | <S\_HASHED\_CC> = MTD\_GENERATE\_HASHED\_CC(#CONFIRMATION\_CODE1, <S\_TRANSACTION\_ID>) | | |
| IC2 | S\_SM-DP+ generates a random <OTPK\_EUICC\_ECKA> | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( #PREP\_DOWNLOAD\_RETRY\_CC) | #R\_PREP\_DOWNLOAD\_WITH\_CC  SW=0x9000  The <EUICC\_SIGNATURE2> SHALL be verified with the #PK\_EUICC\_SIG.  Verify that the <S\_TRANSACTION\_ID> present in the euiccSigned2 is the same as in #PREP\_DOWNLOAD\_RETRY\_CC.  Verify that the <S\_HASHED\_CC> present in the euiccSigned2 is the same as in #PREP\_DOWNLOAD\_RETRY\_CC.  Verify that the <OTPK\_EUICC\_ECKA> present in the euiccSigned2 is not the same as in #PREP\_DOWNLOAD\_RETRY\_CC. |

##### 4.2.10.2.2 TC\_eUICC\_ES10b.PrepareDownloadBRP

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is not loaded on the eUICC. |
| eUICC | The communication between the S\_Device and the eUICC has been initialized and the S\_LPAd has selected the ISD-R.  Common Mutual Authentication procedure has been successfully executed between the eUICC and the S\_SM-DP+   * the same GSMA CI based on BrainpoolP256r1 curve has been chosen for signing and for verification |

Test Sequence #01 Nominal: Without Confirmation Code

This test sequence SHALL be the same as the Test Sequence #01 defined in section 4.2.10.2.1 – TC\_eUICC\_ES10b.PrepareDownloadNIST except that all keys and certificates SHALL be based on BrainpoolP256r1.

Test Sequence #02 Nominal: With Confirmation Code

This test sequence SHALL be the same as the Test Sequence #02 defined in section 4.2.10.2.1 – TC\_eUICC\_ES10b.PrepareDownloadNIST except that all keys and certificates SHALL be based on BrainpoolP256r1.

Test Sequence #03 Nominal: With an unknown otPK.EUICC.ECKA

This test sequence SHALL be the same as the Test Sequence #03 defined in section 4.2.10.2.1 – TC\_eUICC\_ES10b.PrepareDownloadNIST except that all keys and certificates SHALL be based on BrainpoolP256r1.

##### 4.2.10.2.3 TC\_eUICC\_ES10b.PrepareDownloadFRP

This test case is defined as FFS and not applicable for this version of test specification.

##### 4.2.10.2.4 TC\_eUICC\_ES10b.PrepareDownloadErrorCases

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is not loaded on the eUICC. |
| eUICC | The communication between the S\_Device and the eUICC has been initialized and the S\_LPAd has selected the ISD-R.  Common Mutual Authentication procedure has been successfully executed between the eUICC and the S\_SM-DP+ |

Test Sequence #01 Error: VOID

Test Sequence #02 Error: With incorrect CERT.DPpb.SIG (i.e. invalid signature)

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  #PREP\_DOWNLOAD\_INV\_CERT) | #R\_PREP\_DOWNLOAD\_INV\_CERT  SW=0x9000  Verify that the <S\_TRANSACTION\_ID> present in the response is the same as in #PREP\_DOWNLOAD\_INV\_CERT. |

Test Sequence #03 Error: CERT.DPpb.SIG and CERT.DPauth.SIG not belonging to the same entity

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  #PREP\_DOWNLOAD\_CERT\_SMDP2) | #R\_PREP\_DOWNLOAD\_INV\_CERT  SW=0x9000  Verify that the <S\_TRANSACTION\_ID> present in the response is the same as in #PREP\_DOWNLOAD\_CERT\_SMDP2. |

Test Sequence #04 Error: With invalid SM-DP+ signature

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  #PREP\_DOWNLOAD\_INV\_SIGN) | #R\_PREP\_DOWNLOAD\_INV\_SIGN  SW=0x9000  Verify that the <S\_TRANSACTION\_ID> present in the response is the same as in #PREP\_DOWNLOAD\_INV\_SIGN. |

Test Sequence #05 Error: With invalid Transaction ID

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  #PREP\_DOWNLOAD\_INV\_TRANS\_ID) | #R\_PREP\_DOWN\_INV\_TRANS\_ID  SW=0x9000  The transactionId returned in the response SHALL not be checked (any value SHALL be accepted) |

Test Sequence #06 Error: SM-DP+ has not been previously authenticated

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | No Common Mutual Authentication procedure has been executed between the eUICC and the S\_SM-DP+  (this condition overrides the last general initial condition defined in this test case) |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the highest priority CI from <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> and choose #CERT\_S\_SM\_DPpb\_SIG according to this CI curve. |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( #PREP\_DOWNLOAD\_NO\_AUTH) | #R\_PREP\_DOWN\_NO\_SESSION  SW=0x9000  The transactionId returned in the response SHALL not be checked (any value SHALL be accepted) |

Test Sequence #07 Error: Unsupported curve

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( #PREP\_DOWN\_INV\_CURVE) | #R\_PREP\_DOWN\_INV\_CERT  SW=0x9000  Verify that the <S\_TRANSACTION\_ID> present in the response is the same as in #PREP\_DOWN\_INV\_CERT. |

Test Sequence #08 Error: Invalid Certificate Role OID

The purpose of this sequence is to make sure that the eUICC refuses any SM-DP+ Certificate for Profile Package Binding that does not indicate “id-rspRole-dp-pb” in its extension for Certificate Policies.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( #PREP\_DOWNLOAD\_INV\_OID) | #R\_PREP\_DOWNLOAD\_INV\_CERT  SW=0x9000  Verify that the <S\_TRANSACTION\_ID> present in the response is the same as in #PREP\_DOWNLOAD\_INV\_OID. |

### 4.2.11 ES10b (LPA -- eUICC): LoadBoundProfilePackage

#### 4.2.11.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 2.4.3 2.4.5.1
* Section 2.5.3, 2.5.6
* Section 2.6.4, 2.6.5, 2.6.7.1, 2.6.7.2, 2.6.7.3
* Section 3.1.3.2, 3.1.3.3, 3.1.5
* Section 3.2.5
* Section 3.5
* Section 5.5, 5.5.1
* Section 5.7.6
* Annex D
* Annex G

#### 4.2.11.2 Test Cases

##### 4.2.11.2.1 TC\_eUICC\_ES10b.LoadBoundProfilePackageNIST

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is not loaded on the eUICC. |
| eUICC | The communication between the S\_Device and the eUICC has been initialized and the S\_LPAd has selected the ISD-R.  Common Mutual Authentication procedure has been successfully executed between the eUICC and the S\_SM-DP+   * the same GSMA CI based on NIST P-256 curve has been chosen for signing and for verification   Sub-procedure Profile Download and Installation – End User Confirmation has been successfully executed between the eUICC and the S\_SM-DP+   * #PREP\_DOWNLOAD\_NO\_CC has been sent to the eUICC |

Test Sequence #01 Nominal: By using S-ENC and S-MAC

The purpose of this test is to download the PROFILE\_OPERATIONAL1 by using only the session S-ENC and S-MAC keys resulting from key agreement.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_OP\_PROF1,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Split the <BPP> into several segments arrays named:   * <BPP\_SEG\_INIT> * <BPP\_SEG\_A0> * <BPP\_SEG\_A1> * <BPP\_SEG\_A3>   NOTE: In this test sequence, the data resulting of this operation SHALL be composed of the following TLV arrays:   * <BPP\_SEG\_INIT> contains the tag and length fields of the BoundProfilePackage TLV plus the #S\_INIT\_SC\_PROF1 command * <BPP\_SEG\_A0> contains the tag and length fields of the firstSequenceOf87 TLV plus the first 0x87 TLV containing #CONF\_ISDP\_PROF1 command * <BPP\_SEG\_A1> contains the tag and length fields of the sequenceOf88 TLV and each of the '88' TLVs containing #METADATA\_OP\_PROF1 command * <BPP\_SEG\_A3> contains the tag and length fields of the sequenceOf86 TLV and each of the '86' TLVs containing #UPP\_OP\_PROF1 protected with <S\_ENC> and <S\_MAC> | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data #R\_PIR\_OK  for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG. <ISD\_P\_AID> SHALL be in the range as defined SGP.02 [1]. |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  …  iccid #ICCID\_OP\_PROF1,  isdpAid <ISD\_P\_AID>,  profileState disabled,  …  }  }  SW=0x9000 |

Test Sequence #02 Nominal: By using PPK-ENC and PPK-MAC

The purpose of this test is to download the PROFILE\_OPERATIONAL1 by using a new set of random session keys: PPK-ENC, PPK-MAC and Initial MAC chaining value.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_OP\_PROF1,  #REPLACE\_S\_KEYS\_REQ,  #UPP\_OP\_PROF1) | | |
| IC3 | Split the <BPP> into several segments arrays named:   * <BPP\_SEG\_INIT> * <BPP\_SEG\_A0> * <BPP\_SEG\_A1> * <BPP\_SEG\_A2> * <BPP\_SEG\_A3>   NOTE: In this test sequence, the data resulting of this operation SHALL be composed of the following TLV arrays:   * <BPP\_SEG\_INIT> contains the tag and length fields of the BoundProfilePackage TLV plus the #S\_INIT\_SC\_PROF1 command * <BPP\_SEG\_A0> contains the tag and length fields of the firstSequenceOf87 TLV plus the first 0x87 TLV containing #CONF\_ISDP\_PROF1 command * <BPP\_SEG\_A1> contains the tag and length fields of the sequenceOf88 TLV and each of the '88' TLVs containing #METADATA\_OP\_PROF1 command * <BPP\_SEG\_A2> contains the tag and length fields of the secondSequenceOf87 TLV plus the first '87' TLV, containing the #REPLACE\_S\_KEYS\_REQ command * <BPP\_SEG\_A3> contains the tag and length fields of the sequenceOf86 TLV and each of the '86' TLVs containing #UPP\_OP\_PROF1 protected with PPK-ENC and PPK-MAC | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A2>) | SW=0x9000 without response data for all STORE DATA commands |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data #R\_PIR\_OK  for the last STORE DATA command  The euiccSignPIR SHALL be verified with the #PK\_EUICC\_SIG. <ISD\_P\_AID> SHALL be in the range as defined SGP.02 [1]. |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  …  iccid #ICCID\_OP\_PROF1,  isdpAid <ISD\_P\_AID>,  profileState disabled,  …  }  }  SW=0x9000 |

##### 4.2.11.2.2 TC\_eUICC\_ES10b.LoadBoundProfilePackageBRP

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is not loaded on the eUICC. |
| eUICC | The communication between the S\_Device and the eUICC has been initialized and the S\_LPAd has selected the ISD-R.  Common Mutual Authentication procedure has been successfully executed between the eUICC and the S\_SM-DP+   * the same GSMA CI based on BrainpoolP256r1 curve has been chosen for signing and for verification   Sub-procedure Profile Download and Installation – End User Confirmation has been successfully executed between the eUICC and the S\_SM-DP+   * #PREP\_DOWNLOAD\_NO\_CC has been sent to the eUICC |

Test Sequence #01 Nominal: By using S-ENC and S-MAC

This test sequence SHALL be the same as the Test Sequence #01 defined in section 4.2.11.2.1 – TC\_eUICC\_ES10b.LoadBoundProfilePackageNIST except that all keys and certificates SHALL be based on BrainpoolP256r1.

Test Sequence #02 Nominal: By using PPK-ENC and PPK-MAC

This test sequence SHALL be the same as the Test Sequence #02 defined in section 4.2.11.2.1 – TC\_eUICC\_ES10b. LoadBoundProfilePackageNIST except that all keys and certificates SHALL be based on BrainpoolP256r1.

##### 4.2.11.2.3 VOID

##### 4.2.11.2.4 TC\_eUICC\_ES10b.LoadBoundProfilePackage\_ErrorCases

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is not loaded on the eUICC. |
| eUICC | The communication between the S\_Device and the eUICC has been initialized and the S\_LPAd has selected the ISD-R.  Common Mutual Authentication procedure has been successfully executed between the eUICC and the S\_SM-DP+  Sub-procedure Profile Download and Installation – End User Confirmation has been successfully executed between the eUICC and the S\_SM-DP+   * #PREP\_DOWNLOAD\_NO\_CC has been sent to the eUICC |

Test Sequence #01 Error: Unrecognized leading tag in BPP

The purpose of this test is to ensure that the eUICC rejects any BPP segment with an unrecognized leading tag during Profile download. In such case, the eUICC SHALL return a SW of 0x6A88 and SHALL not discard the download session state.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step | Direction | | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_OP\_PROF1,  NO\_PARAM,  #UPP\_OP\_PROF1) | | | |
| IC3 | Split the <BPP> into several segments arrays named:   * <BPP\_SEG\_INIT> * <BPP\_SEG\_A0> * <BPP\_SEG\_A1> * <BPP\_SEG\_A3> | | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  #UNKNOWN\_BPP\_SEGMENT) | | SW=0x6A88 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | | SW=0x9000 without response data for all STORE DATA commands |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data #R\_PIR\_OK for the last STORE DATA command |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  …  iccid #ICCID\_OP\_PROF1,  isdpAid <ISD\_P\_AID>,  profileState disabled,  …  }  }  SW=0x9000 |

Test Sequence #02 Error: GetEUICCChallenge during BPP loading

The purpose of this test is to ensure that the eUICC accepts an ES10b.GetEUICCChallenge request indicating the start of a new RSP session while a BPP is loaded.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | No Notification is stored in the eUICC's Pending Notifications List. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_OP\_PROF1,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Split the <BPP> into several segments arrays named:   * <BPP\_SEG\_INIT> * <BPP\_SEG\_A0> * <BPP\_SEG\_A1> * <BPP\_SEG\_A3> | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| IC6 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A3>) | SW=0x6A88 or 0x6985 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | resp ProfileInfoListResponse ::=  profileInfoListOk :{}  SW=0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |

### 4.2.12 ES10b (LPA -- eUICC): GetEUICCChallenge

#### 4.2.12.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 3.0.1
* Section 5.7.7

#### 4.2.12.2 Test Cases

##### 4.2.12.2.1 TC\_eUICC\_ES10b.GetEUICCChallenge

Test Sequence #01 Nominal

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  <EUICC\_CHALLENGE> received in this step is different to the <EUICC\_CHALLENGE> in Step 1 |

### 4.2.13 ES10b (LPA -- eUICC): GetEUICCInfo

#### 4.2.13.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 4.3
* Section 5.7.8

#### 4.2.13.2 Test Cases

##### 4.2.13.2.1 TC\_eUICC\_ES10b.GetEUICCInfo1

Test Sequence #01 Nominal

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |

Test Sequence #02 Nominal: GetEUICCInfo call after GetEUICCChallenge

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |

Test Sequence #03 Nominal: GetEUICCInfo1 call after AuthenticateServer using Server Variant O certificates

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING> and <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify if they contain at least one same GSMA CI Key ID |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDP\_CHALLENGE> * <S\_SMDP\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING> * Choose the #CERT\_S\_SM\_DPauth\_SIG leading to the same Root CI certificate | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  NO\_PARAM,  #CRL\_LIST,  FALSE  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1)  SW = 0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |

###### Test Sequence #04 Nominal:GetEUICCInfo1 call after AuthenticateServer using V3 Server certificates – Variant A

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3> and <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify if they contain at least one same GSMA CI Key ID |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:  <S\_TRANSACTION\_ID>  <EUICC\_CHALLENGE>  <S\_SMDP\_CHALLENGE>  <S\_SMDP\_SIGNATURE1>  Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3>  Choose the #CERT\_S\_SM\_DPauth\_SIG and the remaining part of the Variant A Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  #CERT\_S\_SM\_DP\_SubCA\_SIG,  #CRL\_LIST,  TRUE  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1)  SW = 0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |

***Test Sequence #05 Nominal: GetEUICCInfo1 call after AuthenticateServer using V3 Server certificates – Variant B***

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3> and <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify if they contain at least one same GSMA CI Key ID |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:  <S\_TRANSACTION\_ID>  <EUICC\_CHALLENGE>  <S\_SMDP\_CHALLENGE>  <S\_SMDP\_SIGNATURE1>  Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3>  Choose the #CERT\_S\_SM\_DPauth\_SIG and the remaining part of the Variant B Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  #CERT\_CI\_SubCA\_SIG,  #CRL\_LIST,  TRUE  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1)  SW = 0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |

***Test Sequence #06 Nominal: GetEUICCInfo1 call after AuthenticateServer using V3 Server certificates – Variant C***

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3> and <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify if they contain at least one same GSMA CI Key ID |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:  <S\_TRANSACTION\_ID>  <EUICC\_CHALLENGE>  <S\_SMDP\_CHALLENGE>  <S\_SMDP\_SIGNATURE1>  Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3>  Choose the #CERT\_S\_SM\_DPauth\_SIG and the remaining part of the Variant C Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  #CERT\_S\_SM\_DP\_SubCAList\_SIG,  #CRL\_LIST,  TRUE  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1)  SW = 0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |

##### 4.2.13.2.2 VOID

##### 4.2.13.2.3 VOID

##### 4.2.13.2.4 TC\_eUICC\_ES10b.GetEUICCInfo2

Test Sequence #01 Nominal: GetEUICCInfo2 call after AuthenticateServer using Server Variant O

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | |
| IC5 | The following inputs are required for Step IC6 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDP\_CHALLENGE> * <S\_SMDP\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING> * Choose the #CERT\_S\_SM\_DPauth\_SIG leading to the same Root CI certificate | | |
| IC6 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  NO\_PARAM,  #CRL\_LIST,  FALSE  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1)  SW = 0x9000 |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO2) | same EUICCInfo2 data object as in Step IC6 (the extCardResource field SHALL be excluded from the comparison)  SW = 0x9000 |

##### 4.2.13.2.5 VOID

##### 4.2.13.2.6 VOID

##### 4.2.13.2.7 VOID

##### 4.2.13.2.8 TC\_eUICC\_ES10b.GetEUICCInfo2\_RSP\_V3.x

Test Sequence #01 Nominal – RSP Version 3.x

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  Verify if:   * <EUICC\_RSP\_CAPABILITY> is present.   #IUT\_RSP\_VERSION\_HIGHEST is equal to 0x030100 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO2) | #R\_EUICC\_INFO2  Verify if:   * #IUT\_RSP\_VERSION\_HIGHEST is equal to 0x030100 * #IUT\_SIMA\_VERSION is equal to 0x020301 or higher but less than 0x030000 * #IUT\_GLOBALPLATFORM\_VERSION is equal to 0x020300 or higher    <EXT\_CARD\_RESOURCE> contains the "number of installed application" value field is set to '00'   * <EUICC\_RSP\_CAPABILITY> contains   o crlSupport set to '0'  o rpmSupport set to '1' if O\_E\_RPM is supported  (otherwise, it SHALL be set to '0')   * additionalProfile set to '1' * deviceInfoExtensibilitySupport set to '1'. * serviceSpecificDataSupport bit set to '1' * hriServerAddressSupport bit set to '1' if and only if O\_E\_HRI\_ADDRESS\_IN\_PM is supported. * serviceProviderMessageSupport bit set to '1'. * lpaProxySupport bit set to '1' if and only if O\_E\_LPA\_PROXY is supported. * enterpriseProfilesSupport bit set to '1' if and only if O\_E\_ENTERPRISE is supported. * serviceDescriptionSupport bit set to '1' if and only if O\_E\_SERVICE\_DESCRIPTION\_IN\_PM is supported. * deviceChangeSupport bit set to '1' if and only if O\_E\_DEVICE\_CHANGE. * estimatedProfileSizeIndicationSupport bit set to '1' if and only if O\_E\_PROFILE\_SIZE\_IN\_PM is supported. * profileSizeInProfilesInfoSupport bit set to '1' if and only if O\_E\_PROFILE\_SIZE\_IN\_PROFILE\_INFO is supported. * crlStaplingV3Support bit set to '1'. * certChainV3VerificationSupport bit set to '1'. * signedSmdsResponseV3Support bit set to '1'. * euiccRspCapInInfo1 bit set to '1' and <EUICC\_RSP\_CAPABILITY> is the same as in Step 1. * osUpdateSupport bit set to '1' if and only if O\_E\_OS\_UPDATE is supported. * cancelForEmptySpnPnSupport bit set to '1'. * updateNotificationConfigurationInfoSupport bit set to '1'. * updateMetadataV3Support bit set to '1'.    #IUT\_UICC\_CAPABILITY contains  o akaMilenage set to '1'  o Either akaTuak128 or akaTuak256 set to '1'   * #IUT\_TRE\_PROPERTIES is present and equal to   { isDiscrete }, { isIntegrated } or { isIntegrated, usesRemoteMemory }   * #IUT\_TRE\_REFERENCE is not present if #IUT\_TRE\_PROPERTIES is equal to { isDiscrete }   SW = 0x9000 |

##### 4.2.13.2.9 TC\_eUICC\_ES10b.GetEUICCInfo2\_RSP\_V3.x\_Integrated\_eUICC

Test Sequence #01 Nominal

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO2) | #R\_EUICC\_INFO2  Verify that:   * The treProperties field contains   + isIntegrated set to ‘1’   and   * + isDiscrete set to ‘0’ * The treProductReference field is present and not empty   SW = 0x9000  NOTE: usesRemoteMemory can be set to either '0' or '1'. |

### 4.2.14 ES10b (LPA -- eUICC): ListNotification

#### 4.2.14.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 2.5.6
* Section 3.1.3.3
* Section 3.5
* Section 5.7.9

#### 4.2.14.2 Test Cases

Throughout all the ListNotification test cases the maximum number of Notifications simultaneously tested has been set as to two as there is not minimum defined in SGP.21 [3] or SGP.22 [2] for the number of Notifications that can be stored by the eUICC.

##### 4.2.14.2.1 TC\_eUICC\_ES10b.ListNotification

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | No Operational Profile is installed on the eUICC. |
| eUICC | No Notifications are stored in the eUICC's Pending Notifications List. |

Test Sequence #01 Nominal: Install Notification

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1. Do not remove both the Notifications. | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_IN1\_IN1\_PIR SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_OMITTED) | #R\_LIST\_NOTIF\_IN1\_IN1\_PIR SW = 0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_NONE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000  OR  #R\_LIST\_NOTIF\_UNDEFINED\_ERROR SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL) | #R\_LIST\_NOTIF\_IN1\_IN1\_PIR SW = 0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ENABLE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DELETE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL\_ENABLE) | #R\_LIST\_NOTIF\_IN1\_IN1\_PIR SW = 0x9000 |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE\_DELETE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 10 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE\_ENABLE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 11 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL\_ENABLE\_DISABLE) | #R\_LIST\_NOTIF\_IN1\_IN1\_PIR SW = 0x9000 |

Test Sequence #02 Nominal: Enable Notification

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1. Remove both the Notifications. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1. Do not remove the Notification. | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_EN1 SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_OMITTED) | #R\_LIST\_NOTIF\_EN1 SW = 0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_NONE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000  OR  #R\_LIST\_NOTIF\_UNDEFINED\_ERROR SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ENABLE) | #R\_LIST\_NOTIF\_EN1 SW = 0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DELETE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL\_ENABLE) | #R\_LIST\_NOTIF\_EN1 SW = 0x9000 |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE\_DELETE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 10 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE\_ENABLE) | #R\_LIST\_NOTIF\_EN1 SW = 0x9000 |
| 11 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL\_ENABLE\_DISABLE) | #R\_LIST\_NOTIF\_EN1 SW = 0x9000 |

Test Sequence #03 Nominal: Disable Notification

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1. Remove both the Notifications. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1. Remove the Notification. | | |
| IC5 | Disable PROFILE\_OPERATIONAL1. Do not remove the Notification. | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DI1 SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_OMITTED) | #R\_LIST\_NOTIF\_DI1 SW = 0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_NONE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000  OR  #R\_LIST\_NOTIF\_UNDEFINED\_ERROR SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ENABLE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE) | #R\_LIST\_NOTIF\_DI1 SW = 0x9000 |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DELETE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL\_ENABLE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE\_DELETE) | #R\_LIST\_NOTIF\_DI1 SW = 0x9000 |
| 10 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE\_ENABLE) | #R\_LIST\_NOTIF\_DI1 SW = 0x9000 |
| 11 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL\_ENABLE\_DISABLE) | #R\_LIST\_NOTIF\_DI1 SW = 0x9000 |

Test Sequence #04 Nominal: Delete Notification

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1. Remove both the Notifications. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1. Remove the Notification. | | |
| IC5 | Disable PROFILE\_OPERATIONAL1. Remove the Notification. | | |
| IC6 | Delete PROFILE\_OPERATIONAL1. Do not remove the Notification. | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DE1 SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_OMITTED) | #R\_LIST\_NOTIF\_DE1 SW = 0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_NONE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000  OR  #R\_LIST\_NOTIF\_UNDEFINED\_ERROR SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ENABLE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DELETE) | #R\_LIST\_NOTIF\_DE1 SW = 0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL\_ENABLE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE\_DELETE) | #R\_LIST\_NOTIF\_DE1 SW = 0x9000 |
| 10 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE\_ENABLE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 11 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ENABLE\_DISABLE\_DELETE) | #R\_LIST\_NOTIF\_DE1 SW = 0x9000 |

Test Sequence #05 Nominal: Two Install Notifications (PIR) with different Notification Address

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1 with #METADATA\_OP\_PROF1\_NO\_INSTALL. Do not remove the Notification. | | |
| IC4 | Install PROFILE\_OPERATIONAL2 with #METADATA\_OP\_PROF2\_NO\_INSTALL.  The default Profile downloading procedure defined in section 2.2.3.1 SHALL be used with the following exceptions:   * #CERT\_S\_SM\_DP2auth\_SIG SHALL be set in MTD\_AUTHENTICATE\_SMDP rather than #CERT\_S\_SM\_DPauth\_SIG * #TEST\_DP\_ADDRESS2 SHALL be set in MTD\_AUTHENTICATE\_SMDP rather than #TEST\_DP\_ADDRESS1 * #CERT\_S\_SM\_DP2pb\_SIG SHALL be set in #PREP\_DOWNLOAD\_NO\_CC rather than #CERT\_S\_SM\_DPpb\_SIG   Do not remove the Notification. | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_IN1\_PIR\_IN2\_PIR SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_OMITTED) | #R\_LIST\_NOTIF\_IN1\_PIR\_IN2\_PIR  SW = 0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_NONE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000  OR  #R\_LIST\_NOTIF\_UNDEFINED\_ERROR SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL) | #R\_LIST\_NOTIF\_IN1\_PIR\_IN2\_PIR SW = 0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ENABLE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DELETE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL\_ENABLE) | #R\_LIST\_NOTIF\_IN1\_PIR\_IN2\_PIR SW = 0x9000 |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE\_DELETE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 10 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE\_ENABLE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 11 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL\_ENABLE\_DISABLE) | #R\_LIST\_NOTIF\_IN1\_PIR\_IN2\_PIR SW = 0x9000 |

Test Sequence #06 Nominal: Install Notification (PIR) and Enable Notification

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1 with #METADATA\_OP\_PROF1\_NO\_INSTALL. Do not remove the Notification. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1. Do not remove the Notification. | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_IN1\_PIR\_EN1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_OMITTED) | #R\_LIST\_NOTIF\_IN1\_PIR\_EN1  SW = 0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_NONE) | #R\_LIST\_NOTIF\_NONE  SW = 0x9000  OR  #R\_LIST\_NOTIF\_UNDEFINED\_ERROR  SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL) | #R\_LIST\_NOTIF\_IN1\_PIR  SW = 0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ENABLE) | #R\_LIST\_NOTIF\_EN1  SW = 0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE) | #R\_LIST\_NOTIF\_NONE  SW = 0x9000 |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DELETE) | #R\_LIST\_NOTIF\_NONE  SW = 0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL\_ENABLE) | #R\_LIST\_NOTIF\_IN1\_PIR\_EN1  SW = 0x9000 |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE\_DELETE) | #R\_LIST\_NOTIF\_NONE  SW = 0x9000 |
| 10 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE\_ENABLE) | #R\_LIST\_NOTIF\_EN1  SW = 0x9000 |
| 11 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL\_ENABLE\_DISABLE) | #R\_LIST\_NOTIF\_IN1\_PIR\_EN1  SW = 0x9000 |

Test Sequence #07 Nominal: Disable and Delete Notifications

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1. Remove both the notifications. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1. Remove the notification | | |
| IC5 | Disable PROFILE\_OPERATIONAL1. Do not remove the notification | | |
| IC6 | Delete PROFILE\_OPERATIONAL1. Do not remove the Notification | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DI1\_DE1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_OMITTED) | #R\_LIST\_NOTIF\_DI1\_DE1  SW = 0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_NONE) | #R\_LIST\_NOTIF\_NONE  SW = 0x9000  OR  #R\_LIST\_NOTIF\_UNDEFINED\_ERROR  SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL) | #R\_LIST\_NOTIF\_NONE  SW = 0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ENABLE) | #R\_LIST\_NOTIF\_NONE  SW = 0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE) | #R\_LIST\_NOTIF\_DI1  SW = 0x9000 |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DELETE) | #R\_LIST\_NOTIF\_DE1  SW = 0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL\_ENABLE) | #R\_LIST\_NOTIF\_NONE  SW = 0x9000 |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE\_DELETE) | #R\_LIST\_NOTIF\_DI1\_DE1  SW = 0x9000 |
| 10 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE\_ENABLE) | #R\_LIST\_NOTIF\_DI1  SW = 0x9000 |
| 11 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL\_ENABLE\_DISABLE) | #R\_LIST\_NOTIF\_DI1  SW = 0x9000 |

Test Sequence #08 Nominal: Install (OtherSignedNotification) and Enable Notification

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1. Remove the PIR notification, but do not remove the OtherSignedNotification. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1. Do not remove the Notification. | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_IN1\_EN1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_OMITTED) | #R\_LIST\_NOTIF\_IN1\_EN1  SW = 0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_NONE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000  OR  #R\_LIST\_NOTIF\_UNDEFINED\_ERROR SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL) | #R\_LIST\_NOTIF\_IN1  SW = 0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ENABLE) | #R\_LIST\_NOTIF\_EN1  SW = 0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE) | #R\_LIST\_NOTIF\_NONE  SW = 0x9000 |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DELETE) | #R\_LIST\_NOTIF\_NONE  SW = 0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL\_ENABLE) | #R\_LIST\_NOTIF\_IN1\_EN1  SW = 0x9000 |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE\_DELETE) | #R\_LIST\_NOTIF\_NONE  SW = 0x9000 |
| 10 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE\_ENABLE) | #R\_LIST\_NOTIF\_EN1  SW = 0x9000 |
| 11 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL\_ENABLE\_DISABLE) | #R\_LIST\_NOTIF\_IN1\_EN1  SW = 0x9000 |

Test Sequence #09 Nominal: Enable and Install (PIR) Notifications

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1. Remove both notifications. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1. Do not remove the Notification. | | |
| IC5 | Install PROFILE\_OPERATIONAL2 with #METADATA\_OP\_PROF2\_NO\_INSTALL.  The default Profile downloading procedure defined in section 2.2.3.1 SHALL be used with the following exceptions:   * #CERT\_S\_SM\_DP2auth\_SIG SHALL be set in MTD\_AUTHENTICATE\_SMDP rather than #CERT\_S\_SM\_DPauth\_SIG * #TEST\_DP\_ADDRESS2 SHALL be set in MTD\_AUTHENTICATE\_SMDP rather than #TEST\_DP\_ADDRESS1 * #CERT\_S\_SM\_DP2pb\_SIG SHALL be set in #PREP\_DOWNLOAD\_NO\_CC rather than #CERT\_S\_SM\_DPpb\_SIG   Do not remove the Notification. | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_EN1\_IN2\_PIR  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_OMITTED) | #R\_LIST\_NOTIF\_EN1\_IN2\_PIR  SW = 0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_NONE) | #R\_LIST\_NOTIF\_NONE  SW = 0x9000  OR  #R\_LIST\_NOTIF\_UNDEFINED\_ERROR  SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL) | #R\_LIST\_NOTIF\_IN2\_PIR  SW = 0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ENABLE) | #R\_LIST\_NOTIF\_EN1  SW = 0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE) | #R\_LIST\_NOTIF\_NONE  SW = 0x9000 |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DELETE) | #R\_LIST\_NOTIF\_NONE  SW = 0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL\_ENABLE) | #R\_LIST\_NOTIF\_EN1\_IN2\_PIR  SW = 0x9000 |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE\_DELETE) | #R\_LIST\_NOTIF\_NONE  SW = 0x9000 |
| 10 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE\_ENABLE) | #R\_LIST\_NOTIF\_EN1  SW = 0x9000 |
| 11 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL\_ENABLE\_DISABLE) | #R\_LIST\_NOTIF\_EN1\_IN2\_PIR  SW = 0x9000 |

Test Sequence #10 Nominal: No Notifications available

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_NONE  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_OMITTED) | #R\_LIST\_NOTIF\_NONE  SW = 0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_NONE) | #R\_LIST\_NOTIF\_NONE  SW = 0x9000  OR  #R\_LIST\_NOTIF\_UNDEFINED\_ERROR  SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL) | #R\_LIST\_NOTIF\_NONE  SW = 0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ENABLE) | #R\_LIST\_NOTIF\_NONE  SW = 0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE) | #R\_LIST\_NOTIF\_NONE  SW = 0x9000 |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DELETE) | #R\_LIST\_NOTIF\_NONE  SW = 0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL\_ENABLE) | #R\_LIST\_NOTIF\_NONE  SW = 0x9000 |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE\_DELETE) | #R\_LIST\_NOTIF\_NONE  SW = 0x9000 |
| 10 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE\_ENABLE) | #R\_LIST\_NOTIF\_NONE  SW = 0x9000 |
| 11 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL\_ENABLE\_DISABLE) | #R\_LIST\_NOTIF\_NONE  SW = 0x9000 |

Test Sequence #11 Nominal: Install Notification, different SM-DP+ Addresses in PIR and Install Notification

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1 with #METADATA\_OP\_PROF1\_INST\_DIFF instead of #METADATA\_OP\_PROF1. Do not remove both the Notifications. | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_IN1\_DP1\_PIR\_IN1\_DP2\_OSN SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_OMITTED) | #R\_LIST\_NOTIF\_IN1\_DP1\_PIR\_IN1\_DP2\_OSN SW = 0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_NONE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000  OR  #R\_LIST\_NOTIF\_UNDEFINED\_ERROR SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL) | #R\_LIST\_NOTIF\_IN1\_DP1\_PIR\_IN1\_DP2\_OSN SW = 0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ENABLE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DELETE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL\_ENABLE) | #R\_LIST\_NOTIF\_IN1\_DP1\_PIR\_IN1\_DP2\_OSN SW = 0x9000 |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE\_DELETE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 10 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE\_ENABLE) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 11 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_INSTALL\_ENABLE\_DISABLE) | #R\_LIST\_NOTIF\_IN1\_DP1\_PIR\_IN1\_DP2\_OSN SW = 0x9000 |

##### 4.2.14.2.2 TC\_eUICC\_ES10b.ListNotification\_RPM

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | No Operational Profile is installed on the eUICC. |
| eUICC | No Notification is stored in the eUICC's Pending Notifications List. |

Test Sequence #01 Nominal: RPM Enable Notification

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1 with #METADATA\_OP\_PROF1\_RPM\_CONF.  Remove the PIR Notification. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1 via RPM as defined in section 2.2.3.4.  Remove the RPR Notification, but do not remove the OtherSignedNotification. | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL\_RPM) | #R\_LIST\_NOTIF\_EN1\_RPM SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_OMITTED) | #R\_LIST\_NOTIF\_EN1\_RPM SW = 0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ENABLE\_RPM) | #R\_LIST\_NOTIF\_EN1\_RPM SW = 0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE\_RPM) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DELETE\_RPM) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE\_DELETE\_RPM) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 10 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE\_ENABLE\_RPM) | #R\_LIST\_NOTIF\_EN1\_RPM SW = 0x9000 |
| 11 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DELETE\_ENABLE\_RPM) | #R\_LIST\_NOTIF\_EN1\_RPM SW = 0x9000 |

Test Sequence #02 Nominal: RPM Enable Notification – RPR and RemoteEnable OSN to different SM-DP+

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1 with #METADATA\_OP\_PROF1\_RPM\_CONF\_OSN\_DP2.  Remove the PIR Notification. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1 via RPM as defined in section 2.2.3.4.  Do not remove the RPR Notification and the OtherSignedNotifications. | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL\_RPM) | #R\_LIST\_NOTIF\_RPR\_EN1\_RPM\_DP2 SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_OMITTED) | #R\_LIST\_NOTIF\_RPR\_EN1\_RPM\_DP2 SW = 0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ENABLE\_RPM) | #R\_LIST\_NOTIF\_EN1\_RPM\_DP2 SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_RPR\_ENABLE\_RPM) | #R\_LIST\_NOTIF\_RPR\_EN1\_RPM\_DP2 SW = 0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_RPR\_DISABLE\_RPM) | #R\_LIST\_NOTIF\_RPR SW = 0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_RPR\_DELETE\_RPM) | #R\_LIST\_NOTIF\_RPR SW = 0x9000 |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_RPR\_DISABLE\_DELETE\_RPM) | #R\_LIST\_NOTIF\_RPR SW = 0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_RPR\_DISABLE\_ENABLE\_RPM) | #R\_LIST\_NOTIF\_RPR\_EN1\_RPM\_DP2 SW = 0x9000 |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_RPR\_DELETE\_ENABLE\_RPM) | #R\_LIST\_NOTIF\_RPR\_EN1\_RPM\_DP2 SW = 0x9000 |
| 10 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_DISABLE\_DELETE\_RPM) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |

### 4.2.15 ES10b (LPA -- eUICC): RetrieveNotificationsList

#### 4.2.15.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 2.5.6
* Section 2.6.7.2
* Section 3.1.3.3
* Section 3.5
* Section 5.7.10

#### 4.2.15.2 Test Cases

Throughout all the RetrieveNotificationsList test cases the maximum number of Notifications simultaneously tested has been set to two as there is no minimum defined in SGP.21 [3] or SGP.22 [2] for the number of Notifications that can be stored by the eUICC.

In some test sequences defined below, it is expected to retrieve especially either a ProfileInstallationResult or an OtherSignedNotification for installation. When both are present in the eUICC, the only way to distinguish these two notifications is to compare their sequence numbers in the ListNotificationResponse. The sequence number related to the ProfileInstallationResult SHALL be lower than the one linked to the OtherSignedNotification.

This assumption is based on the requirement defined in section 5.5.5 of SGP.22 [2] stating that the eUICC SHALL first generate the Profile Installation Result and then as many Notifications as configured in its metadata in the format of OtherSignedNotification.

##### 4.2.15.2.1 TC\_eUICC\_ES10b.RetrieveNotificationsList

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | No Operational Profile is installed on the eUICC. |
| eUICC | No Notifications are stored in the eUICC's Pending Notifications List. |

Test Sequence #01 Nominal: Retrieve by Sequence Number for Install Notification

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1. Do not remove both the notifications. | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_IN1\_IN1\_PIR  SW = 0x9000 |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_RETRIEVE\_NOTIF\_SEQ\_NUM(<NOTIF\_SEQ\_NO\_IN1>)) | #R\_RETRIEVE\_NOTIF\_IN1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_RETRIEVE\_NOTIF\_SEQ\_NUM(<NOTIF\_SEQ\_NO\_IN1\_PIR>)) | #R\_RETRIEVE\_NOTIF\_IN1\_PIR SW = 0x9000  • Verify the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |

Test Sequence #02 Nominal: Retrieve by Sequence Number for Enable Notification

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1. Remove both the notifications. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1. Do not remove the Notification. | | |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_EN1 SW = 0x9000 |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_RETRIEVE\_NOTIF\_SEQ\_NUM(<NOTIF\_SEQ\_NO\_EN1>)) | #R\_RETRIEVE\_NOTIF\_EN1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |

Test Sequence #03 Nominal: Retrieve by Sequence Number for Disable Notification

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1. Remove both the notifications. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1. Remove the Notification. | | |
| IC5 | Disable PROFILE\_OPERATIONAL1. Do not remove the Notification. | | |
| IC6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DI1 SW = 0x9000 |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_RETRIEVE\_NOTIF\_SEQ\_NUM(<NOTIF\_SEQ\_NO\_DI1>)) | #R\_RETRIEVE\_NOTIF\_DI1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |

Test Sequence #04 Nominal: Retrieve by Sequence Number for Delete Notification

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1. Remove both the notifications. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1. Remove the Notification. | | |
| IC5 | Disable PROFILE\_OPERATIONAL1. Remove the Notification. | | |
| IC6 | Delete PROFILE\_OPERATIONAL1. Do not remove the Notification. | | |
| IC7 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DE1 SW = 0x9000 |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_RETRIEVE\_NOTIF\_SEQ\_NUM(<NOTIF\_SEQ\_NO\_DE1>)) | #R\_RETRIEVE\_NOTIF\_DE1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |

Test Sequence #05 Nominal: Retrieve by Sequence Number for Two Install (PIR) Notifications with different Notification Addresses

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1 with #METADATA\_OP\_PROF1\_NO\_INSTALL. Do not remove the Notification. | | |
| IC4 | Install PROFILE\_OPERATIONAL2 with #METADATA\_OP\_PROF2\_NO\_INSTALL.  The default Profile downloading procedure defined in section 2.2.3.1 SHALL be used with the following exceptions:   * #CERT\_S\_SM\_DP2auth\_SIG SHALL be set in MTD\_AUTHENTICATE\_SMDP rather than #CERT\_S\_SM\_DPauth\_SIG * #TEST\_DP\_ADDRESS2 SHALL be set in MTD\_AUTHENTICATE\_SMDP rather than #TEST\_DP\_ADDRESS1 * #CERT\_S\_SM\_DP2pb\_SIG SHALL be set in #PREP\_DOWNLOAD\_NO\_CC rather than #CERT\_S\_SM\_DPpb\_SIG   Do not remove the Notification. | | |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_IN1\_PIR\_IN2\_PIR SW = 0x9000 |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_RETRIEVE\_NOTIF\_SEQ\_NUM(<NOTIF\_SEQ\_NO\_IN1\_PIR>)) | #R\_RETRIEVE\_NOTIF\_IN1\_PIR SW = 0x9000  • Verify the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_RETRIEVE\_NOTIF\_SEQ\_NUM(<NOTIF\_SEQ\_NO\_IN2\_PIR>)) | #R\_RETRIEVE\_NOTIF\_IN2\_PIR SW = 0x9000  • Verify the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |

Test Sequence #06 Nominal: Retrieve by Sequence Number for Install (PIR) and Enable Notifications

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1 with #METADATA\_OP\_PROF1\_NO\_INSTALL. Do not remove the Notification. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1. Do not remove the Notification. | | |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_IN1\_PIR\_EN1 SW = 0x9000 |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_RETRIEVE\_NOTIF\_SEQ\_NUM(<NOTIF\_SEQ\_NO\_IN1\_PIR>)) | #R\_RETRIEVE\_NOTIF\_IN1\_PIR SW = 0x9000  • Verify the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_RETRIEVE\_NOTIF\_SEQ\_NUM(<NOTIF\_SEQ\_NO\_EN1>)) | #R\_RETRIEVE\_NOTIF\_EN1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |

Test Sequence #07 Nominal: Retrieve by Sequence Number for Disable and Delete Notifications

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1. Remove both the notifications. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1. Remove the notification | | |
| IC5 | Disable PROFILE\_OPERATIONAL1. Do not remove the notification | | |
| IC6 | Delete PROFILE\_OPERATIONAL1. Do not remove the Notification | | |
| IC7 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DI1\_DE1 SW = 0x9000 |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_RETRIEVE\_NOTIF\_SEQ\_NUM(<NOTIF\_SEQ\_NO\_DI1>)) | #R\_RETRIEVE\_NOTIF\_DI1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_RETRIEVE\_NOTIF\_SEQ\_NUM(<NOTIF\_SEQ\_NO\_DE1>)) | #R\_RETRIEVE\_NOTIF\_DE1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |

Test Sequence #08 Nominal: Retrieve by Sequence Number for Install (OtherSignedNotification) and Enable Notifications

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1. Remove the PIR notification, but do not remove the OtherSignedNotification. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1. Do not remove the Notification. | | |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_IN1\_EN1 SW = 0x9000 |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_RETRIEVE\_NOTIF\_SEQ\_NUM(<NOTIF\_SEQ\_NO\_IN1>)) | #R\_RETRIEVE\_NOTIF\_IN1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_RETRIEVE\_NOTIF\_SEQ\_NUM(<NOTIF\_SEQ\_NO\_EN1>)) | #R\_RETRIEVE\_NOTIF\_EN1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |

Test Sequence #09 Nominal: Retrieve by Sequence Number for Enable and Install (PIR) notifications

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1. Remove both notifications. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1. Do not remove the Notification. | | |
| IC5 | Install PROFILE\_OPERATIONAL2 with #METADATA\_OP\_PROF2\_NO\_INSTALL.  The default Profile downloading procedure defined in section 2.2.3.1 SHALL be used with the following exceptions:   * #CERT\_S\_SM\_DP2auth\_SIG SHALL be set in MTD\_AUTHENTICATE\_SMDP rather than #CERT\_S\_SM\_DPauth\_SIG * #TEST\_DP\_ADDRESS2 SHALL be set in MTD\_AUTHENTICATE\_SMDP rather than #TEST\_DP\_ADDRESS1 * #CERT\_S\_SM\_DP2pb\_SIG SHALL be set in #PREP\_DOWNLOAD\_NO\_CC rather than #CERT\_S\_SM\_DPpb\_SIG   Do not remove the Notification. | | |
| IC6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_EN1\_IN2\_PIR  SW = 0x9000 |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_RETRIEVE\_NOTIF\_SEQ\_NUM(<NOTIF\_SEQ\_NO\_IN2\_PIR>)) | #R\_RETRIEVE\_NOTIF\_IN2\_PIR SW = 0x9000  • Verify the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_RETRIEVE\_NOTIF\_SEQ\_NUM(<NOTIF\_SEQ\_NO\_EN1>)) | #R\_RETRIEVE\_NOTIF\_EN1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |

Test Sequence #10 Nominal: Retrieve Sequence Numbers that are not present

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1 with #METADATA\_OP\_PROF1\_NO\_INSTALL. Do not remove the Notification. | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_IN1\_PIR SW = 0x9000 |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_RETRIEVE\_NOTIF\_SEQ\_NUM(<NOTIF\_SEQ\_NO\_IN1\_PIR>)) | #R\_RETRIEVE\_NOTIF\_IN1\_PIR SW = 0x9000  • Verify the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_RETRIEVE\_NOTIF\_SEQ\_NUM(<NOTIF\_SEQ\_NO\_IN1\_PIR> +1)) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |

Test Sequence #11 Nominal: Retrieve by Notification Type for Install Notifications

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1. Do not remove both the notifications. | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_ALL) | #R\_RETRIEVE\_NOTIF\_IN1\_IN1\_PIR SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG  • Verify the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_OMITTED) | #R\_RETRIEVE\_NOTIF\_IN1\_IN1\_PIR SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG  • Verify the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_NONE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL) | #R\_RETRIEVE\_NOTIF\_IN1\_IN1\_PIR  SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG  • Verify the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_ENABLE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DELETE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL \_ENABLE) | #R\_RETRIEVE\_NOTIF\_IN1\_IN1\_PIR SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG  • Verify the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE\_DELETE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 10 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE\_ENABLE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 11 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL\_ENABLE\_DISABLE) | #R\_RETRIEVE\_NOTIF\_IN1\_IN1\_PIR SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG  • Verify the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |

Test Sequence #12 Nominal: Retrieve by Notification Type for Enable Notification

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1. Remove both the notifications. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1. Do not remove the Notification. | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_ALL) | #R\_RETRIEVE\_NOTIF\_EN1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_OMITTED) | #R\_RETRIEVE\_NOTIF\_EN1\_V3 SW = 0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_NONE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_ENABLE) | #R\_RETRIEVE\_NOTIF\_EN1\_V3 SW = 0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DELETE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL\_ENABLE) | #R\_RETRIEVE\_NOTIF\_EN1\_V3 SW = 0x9000 |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE\_DELETE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 10 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE\_ENABLE) | #R\_RETRIEVE\_NOTIF\_EN1\_V3 SW = 0x9000 |
| 11 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL\_ENABLE\_DISABLE) | #R\_RETRIEVE\_NOTIF\_EN1\_V3 SW = 0x9000 |

Test Sequence #13 Nominal: Retrieve by Notification Type for Disable Notification

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1. Remove both the notifications. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1. Remove the Notification. | | |
| IC5 | Disable PROFILE\_OPERATIONAL1. Do not remove the Notification. | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_ALL) | #R\_RETRIEVE\_NOTIF\_DI1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_OMITTED) | #R\_RETRIEVE\_NOTIF\_DI1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_NONE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL) | #R\_RETRIEVE\_NOTIF\_NONE  SW = 0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_ENABLE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE) | #R\_RETRIEVE\_NOTIF\_DI1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DELETE) | #R\_RETRIEVE\_NOTIF\_NONE  SW = 0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL\_ENABLE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE\_DELETE) | #R\_RETRIEVE\_NOTIF\_DI1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |
| 10 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE\_ENABLE) | #R\_RETRIEVE\_NOTIF\_DI1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |
| 11 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL\_ENABLE\_DISABLE) | #R\_RETRIEVE\_NOTIF\_DI1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |

Test Sequence #14 Nominal: Retrieve by Notification Type for Delete Notification

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1. Remove both the notifications. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1. Remove the Notification. | | |
| IC5 | Disable PROFILE\_OPERATIONAL1. Remove the Notification. | | |
| IC6 | Delete PROFILE\_OPERATIONAL1. Do not remove the Notification. | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_ALL) | #R\_RETRIEVE\_NOTIF\_DE1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_OMITTED) | #R\_RETRIEVE\_NOTIF\_DE1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_NONE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL) | #R\_RETRIEVE\_NOTIF\_NONE  SW = 0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_ENABLE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE) | #R\_RETRIEVE\_NOTIF\_NONE  SW = 0x9000 |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DELETE) | #R\_RETRIEVE\_NOTIF\_DE1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL\_ENABLE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE\_DELETE) | #R\_RETRIEVE\_NOTIF\_DE1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |
| 10 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE\_ENABLE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 11 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL\_ENABLE\_DISABLE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |

Test Sequence #15 Nominal: Retrieve by Notification Type for Two Install (PIR) Notifications with different Notification Addresses

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1 with #METADATA\_OP\_PROF1\_NO\_INSTALL. Do not remove the Notification. | | |
| IC4 | Install PROFILE\_OPERATIONAL2 with #METADATA\_OP\_PROF2\_NO\_INSTALL.  The default Profile downloading procedure defined in section 2.2.3.1 SHALL be used with the following exceptions:   * #CERT\_S\_SM\_DP2auth\_SIG SHALL be set in MTD\_AUTHENTICATE\_SMDP rather than #CERT\_S\_SM\_DPauth\_SIG * #TEST\_DP\_ADDRESS2 SHALL be set in MTD\_AUTHENTICATE\_SMDP rather than #TEST\_DP\_ADDRESS1 * #CERT\_S\_SM\_DP2pb\_SIG SHALL be set in #PREP\_DOWNLOAD\_NO\_CC rather than #CERT\_S\_SM\_DPpb\_SIG * Do not remove the Notification. | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_ALL) | #R\_RETRIEVE\_NOTIF\_IN1\_PIR\_IN2\_PIR SW = 0x9000  • Verify both the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_OMITTED) | #R\_RETRIEVE\_NOTIF\_IN1\_PIR\_IN2\_PIR SW = 0x9000  • Verify both the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_NONE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL) | #R\_RETRIEVE\_NOTIF\_IN1\_PIR\_IN2\_PIR  SW = 0x9000  • Verify both the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_ENABLE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE) | #R\_RETRIEVE\_NOTIF\_NONE  SW = 0x9000 |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DELETE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL\_ENABLE) | #R\_RETRIEVE\_NOTIF\_IN1\_PIR\_IN2\_PIR SW = 0x9000  • Verify both the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE\_DELETE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 10 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE\_ENABLE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 11 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL\_ENABLE\_DISABLE) | #R\_RETRIEVE\_NOTIF\_IN1\_PIR\_IN2\_PIR SW = 0x9000  • Verify both the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |

Test Sequence #16 Nominal: Retrieve by Notification Type for Install (PIR) and Enable Notifications

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1 with #METADATA\_OP\_PROF1\_NO\_INSTALL. Do not remove the Notification. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1. Do not remove the Notification. | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_ALL) | #R\_RETRIEVE\_NOTIF\_IN1\_PIR\_EN1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG  • Verify the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_OMITTED) | #R\_RETRIEVE\_NOTIF\_IN1\_PIR\_EN1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG  • Verify the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_NONE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL) | #R\_RETRIEVE\_NOTIF\_IN1\_PIR SW = 0x9000  • Verify the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_ENABLE) | #R\_RETRIEVE\_NOTIF\_EN1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DELETE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL\_ENABLE) | #R\_RETRIEVE\_NOTIF\_IN1\_PIR\_EN1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG  • Verify the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE\_DELETE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 10 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE\_ENABLE) | #R\_RETRIEVE\_NOTIF\_EN1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |
| 11 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL\_ENABLE\_DISABLE) | #R\_RETRIEVE\_NOTIF\_IN1\_PIR\_EN1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG  • Verify the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |

Test Sequence #17 Nominal: Retrieve by Notification Type for Disable and Delete Notifications

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1. Remove both the notifications. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1. Remove the notification | | |
| IC5 | Disable PROFILE\_OPERATIONAL1. Do not remove the notification | | |
| IC6 | Delete PROFILE\_OPERATIONAL1. Do not remove the Notification | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_ALL) | #R\_RETRIEVE\_NOTIF\_DI1\_DE1\_V3 SW = 0x9000  • Verify both the euiccNotificationSignatures <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_OMITTED) | #R\_RETRIEVE\_NOTIF\_DI1\_DE1\_V3 SW = 0x9000  • Verify both the euiccNotificationSignatures <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_NONE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_ENABLE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE) | #R\_RETRIEVE\_NOTIF\_DI1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DELETE) | #R\_RETRIEVE\_NOTIF\_DE1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL\_ENABLE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE\_DELETE) | #R\_RETRIEVE\_NOTIF\_DI1\_DE1\_V3 SW = 0x9000  • Verify both the euiccNotificationSignatures <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |
| 10 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE\_ENABLE) | #R\_RETRIEVE\_NOTIF\_DI1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |
| 11 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL\_ENABLE\_DISABLE) | #R\_RETRIEVE\_NOTIF\_DI1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |

Test Sequence #18 Nominal: Retrieve by Notification Type for Install (OtherSignedNotification) and Enable Notifications

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1. Remove the PIR notification, but do not remove the OtherSignedNotification. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1. Do not remove the Notification. | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_ALL) | #R\_RETRIEVE\_NOTIF\_IN1\_EN1\_V3 SW = 0x9000  • Verify both the euiccNotificationSignatures <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_OMITTED) | #R\_RETRIEVE\_NOTIF\_IN1\_EN1\_V3 SW = 0x9000  • Verify both the euiccNotificationSignatures <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_NONE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL) | #R\_RETRIEVE\_NOTIF\_IN1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_ENABLE) | #R\_RETRIEVE\_NOTIF\_EN1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DELETE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL\_ENABLE) | #R\_RETRIEVE\_NOTIF\_IN1\_EN1\_V3 SW = 0x9000  • Verify both the euiccNotificationSignatures <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE\_DELETE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 10 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE\_ENABLE) | #R\_RETRIEVE\_NOTIF\_EN1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |
| 11 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL\_ENABLE\_DISABLE) | #R\_RETRIEVE\_NOTIF\_IN1\_EN1\_V3 SW = 0x9000  • Verify both the euiccNotificationSignatures <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |

Test Sequence #19 Nominal: Retrieve by Notification Type for Enable and Install (PIR) notifications

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1. Remove both notifications. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1. Do not remove the Notification. | | |
| IC5 | Install PROFILE\_OPERATIONAL2 with #METADATA\_OP\_PROF2\_NO\_INSTALL.  The default Profile downloading procedure defined in section 2.2.3.1 SHALL be used with the following exceptions:   * #CERT\_S\_SM\_DP2auth\_SIG SHALL be set in MTD\_AUTHENTICATE\_SMDP rather than #CERT\_S\_SM\_DPauth\_SIG * #TEST\_DP\_ADDRESS2 SHALL be set in MTD\_AUTHENTICATE\_SMDP rather than #TEST\_DP\_ADDRESS1 * #CERT\_S\_SM\_DP2pb\_SIG SHALL be set in #PREP\_DOWNLOAD\_NO\_CC rather than #CERT\_S\_SM\_DPpb\_SIG   Do not remove the Notification. | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_ALL) | #R\_RETRIEVE\_NOTIF\_EN1\_IN2\_PIR\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG  • Verify the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_OMITTED) | #R\_RETRIEVE\_NOTIF\_EN1\_IN2\_PIR\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG  • Verify the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_NONE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL) | #R\_RETRIEVE\_NOTIF\_IN2\_PIR SW = 0x9000  • Verify the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_ENABLE) | #R\_RETRIEVE\_NOTIF\_EN1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DELETE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL\_ENABLE) | #R\_RETRIEVE\_NOTIF\_EN1\_IN2\_PIR\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG  • Verify the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE\_DELETE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 10 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE\_ENABLE) | #R\_RETRIEVE\_NOTIF\_EN1\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |
| 11 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL\_ENABLE\_DISABLE) | #R\_RETRIEVE\_NOTIF\_EN1\_IN2\_PIR\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG  • Verify the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |

Test Sequence #20 Nominal: Retrieve by Notification Type for No Notifications available

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_ALL) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_OMITTED) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_NONE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_ENABLE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DELETE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL\_ENABLE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE\_DELETE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 10 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE\_ENABLE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 11 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL\_ENABLE\_DISABLE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |

Test Sequence #21 Nominal: Retrieve by Sequence Number for Install Notification, different SM-DP+ Addresses in PIR and Install Notification

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1 with #METADATA\_OP\_PROF1\_INST\_DIFF instead of #METADATA\_OP\_PROF1. Do not remove both the notifications. | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_IN1\_DP1\_PIR\_IN1\_DP2\_OSN SW = 0x9000 |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_RETRIEVE\_NOTIF\_SEQ\_NUM(<NOTIF\_SEQ\_NO\_IN1\_DP1>)) | #R\_RETRIEVE\_NOTIF\_IN1\_DP1\_PIR SW = 0x9000  • Verify the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_RETRIEVE\_NOTIF\_SEQ\_NUM(<NOTIF\_SEQ\_NO\_IN1\_DP2>)) | #R\_RETRIEVE\_NOTIF\_IN1\_DP2\_OSN\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG |

Test Sequence #22 Nominal: Retrieve by Notification Type for Install Notifications, different SM-DP+ Addresses in PIR and Install Notification

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1 with #METADATA\_OP\_PROF1\_INST\_DIFF instead of #METADATA\_OP\_PROF1. Do not remove both the notifications. | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_ALL) | #R\_RETRIEVE\_NOTIF\_IN1\_DP1\_PIR\_IN1\_DP2\_OSN\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG  • Verify the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_OMITTED) | #R\_RETRIEVE\_NOTIF\_IN1\_DP1\_PIR\_IN1\_DP2\_OSN\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG  • Verify the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_NONE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL) | #R\_RETRIEVE\_NOTIF\_IN1\_DP1\_PIR\_IN1\_DP2\_OSN\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG  • Verify the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_ENABLE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DELETE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL \_ENABLE) | #R\_RETRIEVE\_NOTIF\_IN1\_DP1\_PIR\_IN1\_DP2\_OSN\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG  • Verify the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE\_DELETE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 10 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_DISABLE\_ENABLE) | #R\_RETRIEVE\_NOTIF\_NONE SW = 0x9000 |
| 11 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #RETRIEVE\_NOTIF\_INSTALL\_ENABLE\_DISABLE) | #R\_RETRIEVE\_NOTIF\_IN1\_DP1\_PIR\_IN1\_DP2\_OSN\_V3 SW = 0x9000  • Verify the euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG> using the #PK\_EUICC\_SIG  • Verify the euiccSignPIR <EUICC\_SIGN\_PIR> using the #PK\_EUICC\_SIG |

### 4.2.16 ES10b (LPA -- eUICC): RemoveNotificationFromList

#### 4.2.16.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 2.5.6
* Section 3.1.3.3
* Section 3.5
* Section 5.7.11

#### 4.2.16.2 Test Cases

Throughout all the RemoveNotificationFromList test cases the maximum number of Notifications simultaneously tested has been set as to two as there is no minimum defined in SGP.21 [3] or SGP.22 [2] for the number of Notifications that can be stored by the eUICC.

The rule specified in section 4.2.15.2 explaining the way to distinguish a ProfileInstallationResult from an OtherSignedNotification for installation also applies for the test cases defined below.

##### 4.2.16.2.1 TC\_eUICC\_ES10b.RemoveNotificationFromList

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | No Operational Profile is installed on the eUICC. |
| eUICC | No Notifications are stored in the eUICC's Pending Notifications List. |

Test Sequence #01 Nominal: Install Notification

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1. Do not remove both the notifications. | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_IN1\_IN1\_PIR SW = 0x9000 |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_REMOVE\_NOTIF(  <NOTIF\_SEQ\_NO\_IN1>)) | #R\_REMOVE\_NOTIF\_OK  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_IN1\_PIR SW = 0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_REMOVE\_NOTIF(  <NOTIF\_SEQ\_NO\_IN1\_PIR>)) | #R\_REMOVE\_NOTIF\_OK  SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |

Test Sequence #02 Nominal: Enable Notification

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1. Remove both the notifications. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1. Do not remove the Notification. | | |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_EN1 SW = 0x9000 |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_REMOVE\_NOTIF(  <NOTIF\_SEQ\_NO\_EN1>)) | #R\_REMOVE\_NOTIF\_OK  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |

Test Sequence #03 Nominal: Disable Notification

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1. Remove both the notifications. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1. Remove the Notification. | | |
| IC5 | Disable PROFILE\_OPERATIONAL1. Do not remove the Notification. | | |
| IC6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DI1 SW = 0x9000 |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_REMOVE\_NOTIF(  <NOTIF\_SEQ\_NO\_DI1>)) | #R\_REMOVE\_NOTIF\_OK  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |

Test Sequence #04 Nominal: Delete Notification

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1. Remove both the notifications. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1. Remove the Notification. | | |
| IC5 | Disable PROFILE\_OPERATIONAL1. Remove the Notification. | | |
| IC6 | Delete PROFILE\_OPERATIONAL1. Do not remove the Notification. | | |
| IC7 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DE1 SW = 0x9000 |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_REMOVE\_NOTIF(  <NOTIF\_SEQ\_NO\_DE1>)) | #R\_REMOVE\_NOTIF\_OK  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |

Test Sequence #05 Nominal: Two Install (PIR) Notifications with different Notification Addresses

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1 with #METADATA\_OP\_PROF1\_NO\_INSTALL. Do not remove the Notification. | | |
| IC4 | Install PROFILE\_OPERATIONAL2 with #METADATA\_OP\_PROF2\_NO\_INSTALL.  The default Profile downloading procedure defined in section 2.2.3.1 SHALL be used with the following exceptions:   * #CERT\_S\_SM\_DP2auth\_SIG SHALL be set in MTD\_AUTHENTICATE\_SMDP rather than #CERT\_S\_SM\_DPauth\_SIG * #TEST\_DP\_ADDRESS2 SHALL be set in MTD\_AUTHENTICATE\_SMDP rather than #TEST\_DP\_ADDRESS1 * #CERT\_S\_SM\_DP2pb\_SIG SHALL be set in #PREP\_DOWNLOAD\_NO\_CC rather than #CERT\_S\_SM\_DPpb\_SIG   Do not remove the Notification. | | |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_IN1\_PIR\_IN2\_PIR SW = 0x9000 |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_REMOVE\_NOTIF(  <NOTIF\_SEQ\_NO\_IN1\_PIR>)) | #R\_REMOVE\_NOTIF\_OK  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_IN2\_PIR SW = 0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_REMOVE\_NOTIF(  <NOTIF\_SEQ\_NO\_IN2\_PIR>)) | #R\_REMOVE\_NOTIF\_OK  SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |

Test Sequence #06 Nominal: Install (PIR) and Enable Notifications

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1 with #METADATA\_OP\_PROF1\_NO\_INSTALL. Do not remove the Notification. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1. Do not remove the Notification. | | |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_IN1\_PIR\_EN1 SW = 0x9000 |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_REMOVE\_NOTIF(  <NOTIF\_SEQ\_NO\_IN1\_PIR>)) | #R\_REMOVE\_NOTIF\_OK  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_EN1 SW = 0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_REMOVE\_NOTIF(  <NOTIF\_SEQ\_NO\_EN1>)) | #R\_REMOVE\_NOTIF\_OK  SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |

Test Sequence #07 Nominal: Disable and Delete Notifications

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1. Remove both the Notifications. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1. Remove the Notification | | |
| IC5 | Disable PROFILE\_OPERATIONAL1. Do not remove the Notification | | |
| IC6 | Delete PROFILE\_OPERATIONAL1. Do not remove the Notification | | |
| IC7 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DI1\_DE1 SW = 0x9000 |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_REMOVE\_NOTIF(  <NOTIF\_SEQ\_NO\_DI1>)) | #R\_REMOVE\_NOTIF\_OK  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DE1  SW = 0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_REMOVE\_NOTIF(  <NOTIF\_SEQ\_NO\_DE1>)) | #R\_REMOVE\_NOTIF\_OK  SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |

Test Sequence #08 Nominal: Install (OtherSignedNotification) and Enable Notifications

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1. Remove the PIR notification, but do not remove the OtherSignedNotification. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1. Do not remove the Notification. | | |
| IC5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_IN1\_EN1 SW = 0x9000 |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_REMOVE\_NOTIF(  <NOTIF\_SEQ\_NO\_IN1>)) | #R\_REMOVE\_NOTIF\_OK  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_EN1 SW = 0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_REMOVE\_NOTIF(  <NOTIF\_SEQ\_NO\_EN1>)) | #R\_REMOVE\_NOTIF\_OK  SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |

Test Sequence #09 Nominal: Enable and Install (PIR) notifications

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1. Remove both notifications. | | |
| IC4 | Enable PROFILE\_OPERATIONAL1. Do not remove the Notification. | | |
| IC5 | Install PROFILE\_OPERATIONAL2 with METADATA\_OP\_PROF2\_NO\_INSTALL.  The default Profile downloading procedure defined in section 2.2.3.1 SHALL be used with the following exceptions:   * #CERT\_S\_SM\_DP2auth\_SIG SHALL be set in MTD\_AUTHENTICATE\_SMDP rather than #CERT\_S\_SM\_DPauth\_SIG * #TEST\_DP\_ADDRESS2 SHALL be set in MTD\_AUTHENTICATE\_SMDP rather than #TEST\_DP\_ADDRESS1 * #CERT\_S\_SM\_DP2pb\_SIG SHALL be set in #PREP\_DOWNLOAD\_NO\_CC rather than #CERT\_S\_SM\_DPpb\_SIG   Do not remove the Notification. | | |
| IC6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_EN1\_IN2\_PIR SW = 0x9000 |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_REMOVE\_NOTIF(  <NOTIF\_SEQ\_NO\_IN2\_PIR>)) | #R\_REMOVE\_NOTIF\_OK SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_EN1 SW = 0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_REMOVE\_NOTIF(  <NOTIF\_SEQ\_NO\_EN1>)) | #R\_REMOVE\_NOTIF\_OK SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |

Test Sequence #10 Nominal: Removing Sequence Numbers that are not present

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL1 with #METADATA\_OP\_PROF1\_NO\_INSTALL. Do not remove the Notification. | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_IN1\_PIR SW = 0x9000 |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_REMOVE\_NOTIF(  <NOTIF\_SEQ\_NO\_IN1\_PIR> - 1)) | #R\_REMOVE\_NOTIF\_NOTHING\_TO\_DELETE SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_IN1\_PIR SW = 0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_REMOVE\_NOTIF(  <NOTIF\_SEQ\_NO\_IN1\_PIR> + 1)) | #R\_REMOVE\_NOTIF\_NOTHING\_TO\_DELETE SW = 0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_IN1\_PIR SW = 0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_REMOVE\_NOTIF(  <NOTIF\_SEQ\_NO\_IN1\_PIR>)) | #R\_REMOVE\_NOTIF\_OK SW = 0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_NONE  SW = 0x9000 |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_REMOVE\_NOTIF(  <NOTIF\_SEQ\_NO\_IN1\_PIR>)) | #R\_REMOVE\_NOTIF\_NOTHING\_TO\_DELETE SW = 0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_REMOVE\_NOTIF(  <NOTIF\_SEQ\_NO\_IN1\_PIR> + 1)) | #R\_REMOVE\_NOTIF\_NOTHING\_TO\_DELETE SW = 0x9000 |
| 10 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_NONE SW = 0x9000 |

### 4.2.17 ES10b (LPA -- eUICC): LoadCRL

This section is defined as FFS.

### 4.2.18 ES10b (LPA -- eUICC): AuthenticateServer

#### 4.2.18.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 2.4.2
* Section 2.6.2, 2.6.7.1, 2.6.7.2
* Section 3.0.1, 3.1.3
* Section 3.6.2
* Section 4.5.2
* Section 5.5
* Section 5.6.1
* Section 5.7.13
* Section 5.8.1

#### 4.2.18.2 Test Cases

##### 4.2.18.2.1 TC\_eUICC\_ES10b.AuthenticateServer\_SM-DP+\_NIST\_Server\_Variant\_O

Test Sequence #01 Nominal: Without MatchingID in CtxParams1

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING> and <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify if they contain at least one same GSMA CI Key ID based on NIST P-256 curve |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDP\_CHALLENGE> * <S\_SMDP\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> to the CI Key ID based on NIST P-256 curve * Choose the #CERT\_S\_SM\_DPauth\_SIG leading to the same Root CI certificate | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  NO\_PARAM,  #CRL\_LIST,  FALSE  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1)  SW = 0x9000 |

Test Sequence #02 Nominal: With MatchingID in CtxParams1

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING> and <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify if they contain at least one same GSMA CI Key ID based on NIST P-256 curve |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDP\_CHALLENGE> * <S\_SMDP\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> to the CI Key ID based on NIST P-256 curve * Choose the #CERT\_S\_SM\_DPauth\_SIG leading to the same Root CI certificate | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_MATCH\_ID,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  NO\_PARAM,  #CRL\_LIST,  FALSE  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_MATCH\_ID  )  SW = 0x9000 |

Test Sequence #03 Nominal: With IMEI in Device Capabilities

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING> and <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify if they contain at least one same GSMA CI Key ID based on NIST P-256 curve |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDP\_CHALLENGE> * <S\_SMDP\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> to the CI Key ID based on NIST P-256 curve * Choose the #CERT\_S\_SM\_DPauth\_SIG leading to the same Root CI certificate | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_IMEI,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  NO\_PARAM,  #CRL\_LIST,  FALSE  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_IMEI  )  SW = 0x9000 |

##### 4.2.18.2.2 TC\_eUICC\_ES10b.AuthenticateServer\_SM-DP+\_BRP\_Server\_Variant\_O

Test Sequence #01 Nominal: Without MatchingID in CtxParams1

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING> and <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify if they contain at least one same GSMA CI Key ID based on BrainpoolP256r1 curve |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDP\_CHALLENGE> * <S\_SMDP\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> to the CI Key ID based on BrainpoolP256r1 curve * Choose the #CERT\_S\_SM\_DPauth\_SIG leading to the same Root CI certificate | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  NO\_PARAM,  #CRL\_LIST,  FALSE  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1)  SW = 0x9000 |

Test Sequence #02 Nominal: With MatchingID in CtxParams1

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING> and <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify if they contain at least one same GSMA CI Key ID based on BrainpoolP256r1 curve |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDP\_CHALLENGE> * <S\_SMDP\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> to the CI Key ID based on BrainpoolP256r1 curve * Choose the #CERT\_S\_SM\_DPauth\_SIG leading to the same Root CI certificate | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_MATCH\_ID,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  NO\_PARAM,  #CRL\_LIST,  FALSE  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_MATCH\_ID  )  SW = 0x9000 |

Test Sequence #03 Nominal: With IMEI in Device Capabilities

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING> and <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify if they contain at least one same GSMA CI Key ID based on BrainpoolP256r1 curve |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDP\_CHALLENGE> * <S\_SMDP\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> to the CI Key ID based on BrainpoolP256r1 curve * Choose the #CERT\_S\_SM\_DPauth\_SIG leading to the same Root CI certificate | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_IMEI,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  NO\_PARAM,  #CRL\_LIST,  FALSE  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_IMEI  )  SW = 0x9000 |

##### 4.2.18.2.3 TC\_eUICC\_ES10b.AuthenticateServer\_SM-DP+\_FRP

This test case is defined as FFS and not applicable for this version of test specification.

##### 4.2.18.2.4 TC\_eUICC\_ES10b.AuthenticateServer\_SM-DP+\_ErrorCases\_Server\_Variant\_O

Test Sequence #01 Error: With Incorrect SM-DPauth certificate (i.e. invalid signature)

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   <S\_TRANSACTION\_ID>   <EUICC\_CHALLENGE>   <S\_SMDP\_CHALLENGE>   <S\_SMDP\_SIGNATURE1>   Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING>   Choose the #CERT\_S\_SM\_DPauth\_INV\_SIGN leading to the same Root CI certificate apart from the signature | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT (MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_INV\_SIG,  NO\_PARAM,  #CRL\_LIST,  FALSE  )  ) | #R\_AUTH\_SERVER\_INV\_CERT  SW = 0x9000  • Verify that the <S\_TRANSACTION\_ID> present in the AuthenticateResponseError is the same as <S\_TRANSACTION\_ID> in MTD\_AUTHENTICATE\_SMDP. |

Test Sequence #02 Error: With Invalid SM-DP+ Signature

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   <S\_TRANSACTION\_ID>   <EUICC\_CHALLENGE>   <S\_SMDP\_CHALLENGE>   <S\_SMDP\_SIGNATURE1> NOT computed with the #SK\_S\_SM\_DPauth\_SIG but SHALL have the same length as for a valid signature   Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING>   Choose the #CERT\_S\_SM\_DPauth\_SIG leading to the same Root CI certificate | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  NO\_PARAM,  #CRL\_LIST,  FALSE  )  ) | #R\_AUTH\_SERVER\_INV\_SIGN  SW = 0x9000  • Verify that the <S\_TRANSACTION\_ID> present in the AuthenticateResponseError is the same as <S\_TRANSACTION\_ID> in MTD\_AUTHENTICATE\_SMDP |

Test Sequence #03 Error: Unsupported Curve

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   <S\_TRANSACTION\_ID>   <EUICC\_CHALLENGE>   <S\_SMDP\_CHALLENGE>   <RANDOM\_SM\_DP+\_SIGN>   Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING>   #CERT\_S\_SM\_DPauth\_INV\_CURVE | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT (MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_INV\_CURVE,  NO\_PARAM,  #CRL\_LIST,  FALSE  )  ) | #R\_AUTH\_SERVER\_INV\_CURV  SW = 0x9000  • Verify that the <S\_TRANSACTION\_ID> present in the AuthenticateResponseError is the same as <S\_TRANSACTION\_ID> in MTD\_AUTHENTICATE\_SMDP. |

Test Sequence #04 Error: eUICC Challenge Mismatch

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000 |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   <S\_TRANSACTION\_ID>   Change the value of <EUICC\_CHALLENGE> (retrieved in step 1) to a random value different from <EUICC\_CHALLENGE> <S\_SMDP\_CHALLENGE>   <S\_SMDP\_SIGNATURE1>   Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING>   Choose the #CERT\_S\_SM\_DPauth\_SIG leading to the same Root CI certificate | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT (MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  NO\_PARAM,  #CRL\_LIST,  FALSE  )  ) | #R\_AUTH\_SERVER\_INV\_CHALLENGE  SW = 0x9000  • Verify that the <S\_TRANSACTION\_ID> present in the AuthenticateResponseError is the same as <S\_TRANSACTION\_ID> in MTD\_AUTHENTICATE\_SMDP. |

Test Sequence #05 Error: Unknown CI PK

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   <S\_TRANSACTION\_ID>   <EUICC\_CHALLENGE>   <S\_SMDP\_CHALLENGE>   <S\_SMDP\_SIGNATURE1>   Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> to a CI Key ID not present in the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING> (a random SubjectKeyIdentifier can be used)   Choose the #CERT\_S\_SM\_DPauth\_SIG leading to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT (MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  NO\_PARAM,  #CRL\_LIST,  FALSE  )  ) | #R\_AUTH\_SERVER\_INV\_CI  SW = 0x9000  • Verify that the <S\_TRANSACTION\_ID> present in the AuthenticateResponseError is the same as <S\_TRANSACTION\_ID> in MTD\_AUTHENTICATE\_SMDP. |

Test Sequence #06 Error: Invalid Certificate Role OID

The purpose of this sequence is to make sure that the eUICC refuses any SM-DP+ Certificate for authentication that does not indicate “id-rspRole-dp-auth” in its extension for Certificate Policies.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   <S\_TRANSACTION\_ID>   <EUICC\_CHALLENGE>   <S\_SMDP\_CHALLENGE>   <S\_SMDP\_SIGNATURE1>   Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING>   Choose the #CERT\_S\_SM\_DPpb\_SIG (instead of #CERT\_S\_SM\_DPauth\_SIG) leading to the same Root CI certificate | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT (MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPpb\_SIG,  NO\_PARAM,  #CRL\_LIST,  FALSE  )  ) | #R\_AUTH\_SERVER\_INV\_OID  SW = 0x9000  OR  #R\_AUTH\_SERVER\_INV\_CERT  SW = 0x9000  • Verify that the <S\_TRANSACTION\_ID> present in the AuthenticateResponseError is the same as <S\_TRANSACTION\_ID> in MTD\_AUTHENTICATE\_SMDP. |

Test Sequence #07 Error: No RSP session on-going

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial state |
| eUICC | No RSP session is on-going (i.e. no ES10b.getEUICCChallenge has been sent to the eUICC). |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | The following inputs are required for Step 3 as described in the InitiateAuthentication function:   <S\_TRANSACTION\_ID>   Change the value of <EUICC\_CHALLENGE> (retrieved in step 1) to a random value different from <EUICC\_CHALLENGE>   <S\_SMDP\_CHALLENGE>   <S\_SMDP\_SIGNATURE1>   Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING>   Choose the #CERT\_S\_SM\_DPauth\_SIG leading to the same Root CI certificate | | |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT (MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  NO\_PARAM,  #CRL\_LIST,  FALSE  )  ) | #R\_AUTH\_SERVER\_NO\_SESSION  SW = 0x9000  The transactionId returned in the response SHALL not be checked (any value SHALL be accepted) |

##### 4.2.18.2.5 TC\_eUICC\_ES10b.AuthenticateServer\_SM-DS\_BRP\_Server\_Variant\_O

Test Sequence #01 Nominal: With EventID in CtxParams1

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING> and <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify if they contain at least one same GSMA CI Key ID based on BrainpoolP256r1 curve |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDS\_CHALLENGE> * <S\_SMDS\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> to the CI Key ID based on BrainpoolP256r1 curve * Choose the #CERT\_S\_SM\_DSauth\_SIG leading to the same Root CI certificate | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDS(  #TEST\_ROOT\_DS\_ADDRESS,  <S\_SMDS\_CHALLENGE>,  #CTX\_PARAMS1\_EVENT\_ID,  <S\_SMDS\_SIGNATURE1>,  #CERT\_S\_SM\_DSauth\_SIG,  NO\_PARAM,  #CRL\_LIST,  FALSE  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_ROOT\_DS\_ADDRESS,  <S\_SMDS\_CHALLENGE>,  #CTX\_PARAMS1\_EVENT\_ID)  SW = 0x9000 |

Test Sequence #02 Nominal: With IMEI and EventID in Device Capabilities

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING> and <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify if they contain at least one same GSMA CI Key ID based on BrainpoolP256r1 curve |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDS\_CHALLENGE> * <S\_SMDS\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> to the CI Key ID based on BrainpoolP256r1 curve * Choose the #CERT\_S\_SM\_DSauth\_SIG leading to the same Root CI certificate | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDS(  #TEST\_ROOT\_DS\_ADDRESS,  <S\_SMDS\_CHALLENGE>,  #CTX\_PARAMS1\_EVENT\_ID\_IMEI,  <S\_SMDS\_SIGNATURE1>,  #CERT\_S\_SM\_DSauth\_SIG,  NO\_PARAM,  #CRL\_LIST,  FALSE  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_ROOT\_DS\_ADDRESS,  <S\_SMDS\_CHALLENGE>,  #CTX\_PARAMS1\_EVENT\_ID\_IMEI)  SW = 0x9000 |

##### 4.2.18.2.6 TC\_eUICC\_ES10b.AuthenticateServer\_SM-DS\_NIST\_Server\_Variant\_O

Test Sequence #01 Nominal: With EventID in CtxParams1

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING> and <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify if they contain at least one same GSMA CI Key ID based on NIST P-256 curve |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDS\_CHALLENGE> * <S\_SMDS\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> to the CI Key ID based on NIST P-256 curve * Choose the #CERT\_S\_SM\_DSauth\_SIG leading to the same Root CI certificate | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDS(  #TEST\_ROOT\_DS\_ADDRESS,  <S\_SMDS\_CHALLENGE>,  #CTX\_PARAMS1\_EVENT\_ID,  <S\_SMDS\_SIGNATURE1>,  #CERT\_S\_SM\_DSauth\_SIG,  NO\_PARAM,  #CRL\_LIST,  FALSE  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_ROOT\_DS\_ADDRESS,  <S\_SMDS\_CHALLENGE>,  #CTX\_PARAMS1\_EVENT\_ID)  SW = 0x9000 |

Test Sequence #02 Nominal: With IMEI and EventID in Device Capabilities

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING> and <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify if they contain at least one same GSMA CI Key ID based on NIST P-256 curve |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDS\_CHALLENGE> * <S\_SMDS\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> to the CI Key ID based on NIST P-256 curve * Choose the #CERT\_S\_SM\_DSauth\_SIG leading to the same Root CI certificate | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(MTD\_AUTHENTICATE\_SMDS(  #TEST\_ROOT\_DS\_ADDRESS,  <S\_SMDS\_CHALLENGE>,  #CTX\_PARAMS1\_EVENT\_ID\_IMEI,  <S\_SMDS\_SIGNATURE1>,  #CERT\_S\_SM\_DSauth\_SIG,  NO\_PARAM,  #CRL\_LIST,  FALSE  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_ROOT\_DS\_ADDRESS,  <S\_SMDS\_CHALLENGE>,  #CTX\_PARAMS1\_EVENT\_ID\_IMEI)  SW = 0x9000 |

##### 4.2.18.2.7 TC\_eUICC\_ES10b.AuthenticateServer\_SM-DS\_FRP

This test case is defined as FFS and not applicable for this version of test specification.

##### 4.2.18.2.8 TC\_eUICC\_ES10b.AuthenticateServer\_SM-DS\_ErrorCases\_Server\_Variant\_O

Test Sequence #01 Error: With Incorrect SM-DSauth certificate (i.e. invalid signature)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result | |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 | |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> | |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   <S\_TRANSACTION\_ID>   <EUICC\_CHALLENGE>   <S\_SMDS\_CHALLENGE>   <S\_SMDS\_SIGNATURE1>   Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING>   Choose the #CERT\_S\_SM\_DSauth\_INV\_SIGN leading to the same Root CI certificate | | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDS(  #TEST\_ROOT\_DS\_ADDRESS,  <S\_SMDS\_CHALLENGE>,  #CTX\_PARAMS1\_EVENT\_ID,  <S\_SMDS\_SIGNATURE1>,  #CERT\_S\_SM\_DSauth\_INV\_SIG,  NO\_PARAM,  #CRL\_LIST,  FALSE  )  ) | | #R\_AUTH\_SERVER\_INV\_CERT  SW = 0x9000  • Verify that the <S\_TRANSACTION\_ID> present in the AuthenticateResponseError is the same as <S\_TRANSACTION\_ID> in MTD\_AUTHENTICATE\_SMDS. |

Test Sequence #02 Error: With Invalid SM-DS Signature

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDS\_CHALLENGE> * <S\_SMDS\_SIGNATURE1> NOT computed with the #SK\_S\_SM\_DSauth\_SIG but SHALL have the same length as for a valid signature * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING> * Choose the #CERT\_S\_SM\_DSauth\_SIG leading to the same Root CI certificate | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDS(  #TEST\_ROOT\_DS\_ADDRESS,  <S\_SMDS\_CHALLENGE>,  #CTX\_PARAMS1\_EVENT\_ID,  <S\_SMDS\_SIGNATURE1>,  #CERT\_S\_SM\_DSauth\_SIG,  NO\_PARAM,  #CRL\_LIST,  FALSE  )  ) | #R\_AUTH\_SERVER\_INV\_SIGN  SW = 0x9000  • Verify that the <S\_TRANSACTION\_ID> present in the AuthenticateResponseError is the same as <S\_TRANSACTION\_ID> in MTD\_AUTHENTICATE\_SMDS |

Test Sequence #03 Error: Unsupported Curve

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   <S\_TRANSACTION\_ID>   <EUICC\_CHALLENGE>   <S\_SMDS\_CHALLENGE>   <RANDOM\_SM\_DS\_SIGN>   Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING>   #CERT\_S\_SM\_DSauth\_INV\_CURVE | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT (MTD\_AUTHENTICATE\_SMDS(  #TEST\_ROOT\_DS\_ADDRESS,  <S\_SMDS\_CHALLENGE>,  #CTX\_PARAMS1\_EVENT\_ID,  <S\_SMDS\_SIGNATURE1>,  #CERT\_S\_SM\_DSauth\_INV\_CURVE,  NO\_PARAM,  #CRL\_LIST,  FALSE  )  ) | #R\_AUTH\_SERVER\_INV\_CURV  SW = 0x9000  • Verify that the <S\_TRANSACTION\_ID> present in the AuthenticateResponseError is the same as <S\_TRANSACTION\_ID> in MTD\_AUTHENTICATE\_SMDS. |

Test Sequence #04 Error: eUICC Challenge Mismatch

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000 |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * Change the value of <EUICC\_CHALLENGE> (retrieved in step 1) to a random value different from <EUICC\_CHALLENGE> * <S\_SMDS\_CHALLENGE> * <S\_SMDS\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING> * Choose the #CERT\_S\_SM\_DSauth\_SIG leading to the same Root CI certificate | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT (  MTD\_AUTHENTICATE\_SMDS(  #TEST\_ROOT\_DS\_ADDRESS,  <S\_SMDS\_CHALLENGE>,  #CTX\_PARAMS1\_EVENT\_ID,  <S\_SMDS\_SIGNATURE1>,  #CERT\_S\_SM\_DSauth\_SIG,  NO\_PARAM,  #CRL\_LIST,  FALSE  )  ) | #R\_AUTH\_SERVER\_INV\_CHALLENGE  SW = 0x9000  • Verify that the <S\_TRANSACTION\_ID> present in the AuthenticateResponseError is the same as <S\_TRANSACTION\_ID> in MTD\_AUTHENTICATE\_SMDS . |

Test Sequence #05 Error: Unknown CI PK

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDS\_CHALLENGE> * <S\_SMDS\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> to a CI Key ID not present in the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING> (a random SubjectKeyIdentifier can be used) * Choose the #CERT\_S\_SM\_DSauth\_SIG leading to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT (MTD\_AUTHENTICATE\_SMDS(  #TEST\_ROOT\_DS\_ADDRESS,  <S\_SMDS\_CHALLENGE>,  #CTX\_PARAMS1\_EVENT\_ID,  <S\_SMDS\_SIGNATURE1>,  #CERT\_S\_SM\_DSauth\_SIG,  NO\_PARAM,  #CRL\_LIST,  FALSE  )  ) | #R\_AUTH\_SERVER\_INV\_CI  SW = 0x9000  • Verify that the <S\_TRANSACTION\_ID> present in the AuthenticateResponseError is the same as <S\_TRANSACTION\_ID> in MTD\_AUTHENTICATE\_SMDS . |

Test Sequence #06 Error: No RSP session on-going

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial state |
| eUICC | No RSP session is on-going (i.e. no ES10b.getEUICCChallenge has been sent to the eUICC). |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | The following inputs are required for Step 3 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * Change the value of <EUICC\_CHALLENGE> (retrieved in step 1) to a random value different from <EUICC\_CHALLENGE> * <S\_SMDS\_CHALLENGE> * <S\_SMDS\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING> * Choose the #CERT\_S\_SM\_DSauth\_SIG leading to the same Root CI certificate | | |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDS(  #TEST\_ROOT\_DS\_ADDRESS,  <S\_SMDS\_CHALLENGE>,  #CTX\_PARAMS1\_EVENT\_ID,  <S\_SMDS\_SIGNATURE1>,  #CERT\_S\_SM\_DSauth\_SIG,  NO\_PARAM,  #CRL\_LIST,  FALSE  )  ) | #R\_AUTH\_SERVER\_NO\_SESSION  SW = 0x9000  The transactionId returned in the response SHALL not be checked (any value SHALL be accepted) |

##### 4.2.18.2.9 TC\_eUICC\_ES10b.AuthenticateServer\_SM-DP+\_NIST\_Server\_Variants\_V3

Test Sequence #01 Nominal: Server Variant A without MatchingID in CtxParams1

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3> and <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify if they contain at least one same GSMA CI Key ID based on NIST P-256 curve |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDP\_CHALLENGE> * <S\_SMDP\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID based on NIST P-256 curve * Choose the #CERT\_S\_SM\_DPauth\_SIG and the remaining part of the Variant A Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMPD (  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  #CERT\_S\_SM\_DP\_SubCA\_SIG,  #CRL\_LIST,  TRUE  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1)  SW = 0x9000 |

Test Sequence #02 Nominal: Server Variant A with MatchingID in CtxParams1

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3> and <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify if they contain at least one same GSMA CI Key ID based on NIST P-256 curve |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDP\_CHALLENGE> * <S\_SMDP\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID based on NIST P-256 curve * Choose the #CERT\_S\_SM\_DPauth\_SIG and the remaining part of the Variant A Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_MATCH\_ID,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  #CERT\_S\_SM\_DP\_SubCA\_SIG,  #CRL\_LIST,  TRUE  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_MATCH\_ID)  SW = 0x9000 |

Test Sequence #03 Nominal: Server Variant A with IMEI in Device Capabilities

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3> and <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify if they contain at least one same GSMA CI Key ID based on NIST P-256 curve |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDP\_CHALLENGE> * <S\_SMDP\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID based on NIST P-256 curve * Choose the #CERT\_S\_SM\_DPauth\_SIG and the remaining part of the Variant A Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_IMEI,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  #CERT\_S\_SM\_DP\_SubCA\_SIG,  #CRL\_LIST,  TRUE  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_IMEI)  SW = 0x9000 |

Test Sequence #04 Nominal: Server Variant B without MatchingID in CtxParams1

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3> and <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify if they contain at least one same GSMA CI Key ID based on NIST P-256 curve |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDP\_CHALLENGE> * <S\_SMDP\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID based on NIST P-256 curve * Choose the #CERT\_S\_SM\_DPauth\_SIG and the remaining part of the Variant B Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  #CERT\_CI\_SubCA\_SIG,  #CRL\_LIST,  TRUE  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1)  SW = 0x9000 |

***Test Sequence #05 Nominal: Server Variant B with MatchingID in CtxParams1***

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3> and <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify if they contain at least one same GSMA CI Key ID based on NIST P-256 curve |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:  <S\_TRANSACTION\_ID>  <EUICC\_CHALLENGE>  <S\_SMDP\_CHALLENGE>  <S\_SMDP\_SIGNATURE1>  Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID based on NIST P-256 curve  Choose the #CERT\_S\_SM\_DPauth\_SIG and the remaining part of the Variant B Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_MATCH\_ID,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  #CERT\_CI\_SubCA\_SIG,  #CRL\_LIST,  TRUE  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_MATCH\_ID)  SW = 0x9000 |

***Test Sequence #06 Nominal: Server Variant B with IMEI in Device Capabilities***

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3> and <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify if they contain at least one same GSMA CI Key ID based on NIST P-256 curve |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:  <S\_TRANSACTION\_ID>  <EUICC\_CHALLENGE>  <S\_SMDP\_CHALLENGE>  <S\_SMDP\_SIGNATURE1>  Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID based on NIST P-256 curve  Choose the #CERT\_S\_SM\_DPauth\_SIG and the remaining part of the Variant B Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_IMEI,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  #CERT\_CI\_SubCA\_SIG,  #CRL\_LIST,  TRUE  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_IMEI)  SW = 0x9000 |

Test Sequence #07 Nominal: Server Variant C without MatchingID in CtxParams1

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3> and <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify if they contain at least one same GSMA CI Key ID based on NIST P-256 curve |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDP\_CHALLENGE> * <S\_SMDP\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID based on NIST P-256 curve * Choose the #CERT\_S\_SM\_DPauth\_SIG and the remaining part of the Variant C Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  #CERT\_S\_SM\_DP\_SubCAList\_SIG,  #CRL\_LIST,  TRUE  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1)  SW = 0x9000 |

***Test Sequence #08 Nominal: Server Variant C with MatchingID in CtxParams1***

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3> and <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify if they contain at least one same GSMA CI Key ID based on NIST P-256 curve |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:  <S\_TRANSACTION\_ID>  <EUICC\_CHALLENGE>  <S\_SMDP\_CHALLENGE>  <S\_SMDP\_SIGNATURE1>  Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID based on NIST P-256 curve  Choose the #CERT\_S\_SM\_DPauth\_SIG and the remaining part of the Variant C Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_MATCH\_ID,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  #CERT\_S\_SM\_DP\_SubCAList\_SIG,  #CRL\_LIST,  TRUE  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_MATCH\_ID)  SW = 0x9000 |

***Test Sequence #09 Nominal: Server Variant C with IMEI in Device Capabilities***

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3> and <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify if they contain at least one same GSMA CI Key ID based on NIST P-256 curve |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:  <S\_TRANSACTION\_ID>  <EUICC\_CHALLENGE>  <S\_SMDP\_CHALLENGE>  <S\_SMDP\_SIGNATURE1>  Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID based on NIST P-256 curve  Choose the #CERT\_S\_SM\_DPauth\_SIG and the remaining part of the Variant C Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_IMEI,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  #CERT\_S\_SM\_DP\_SubCAList\_SIG,  #CRL\_LIST,  TRUE  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_IMEI)  SW = 0x9000 |

Test Sequence #10 Nominal: Server Variant A without MatchingID and OperationType RPM in CtxParams1

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3> and <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify if they contain at least one same GSMA CI Key ID based on NIST P-256 curve |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   <S\_TRANSACTION\_ID>   <EUICC\_CHALLENGE>   <S\_SMDP\_CHALLENGE>   <S\_SMDP\_SIGNATURE1>   Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID based on NIST P-256 curve   Choose the #CERT\_S\_SM\_DPauth\_SIG and the remaining part of the Variant A Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_RPM\_ICCID1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  #CERT\_S\_SM\_DP\_SubCA\_ECDSA,  #CRL\_LIST,  TRUE  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_RPM\_ICCID1)  SW = 0x9000 |

##### 4.2.18.2.10 TC\_eUICC\_ES10b.AuthenticateServer\_SM-DP+\_ErrorCases\_V3\_Server\_Variant A

In order to execute the error cases defined in this section, the variant A of the SM-DP+ Certificates chain is used by default.

Test Sequence #01 Error: With Incorrect SM-DPauth certificate (i.e. invalid signature)

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDP\_CHALLENGE> * <S\_SMDP\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3> * Choose the #CERT\_S\_SM\_DPauth\_INV\_SIGN and the remaining part of the Variant A Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_INV\_SIGN,  #CERT\_S\_SM\_DP\_SubCA\_SIG,  #CRL\_LIST,  TRUE  )  ) | MTD\_CHECK\_AUTH\_ERROR\_RESP(  invalidCertificate)  SW = 0x9000 |

Test Sequence #02 Error: With Invalid SM-DP+ Signature

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDP\_CHALLENGE> * <S\_SMDP\_SIGNATURE1> NOT computed with the #SK\_S\_SM\_DPauth\_SIG but SHALL have the same length as for a valid signature * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3> * Choose the #CERT\_S\_SM\_DPauth\_SIG and the remaining part of the Variant A Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  #CERT\_S\_SM\_DP\_SubCA\_SIG,  #CRL\_LIST,  TRUE  )  ) | MTD\_CHECK\_AUTH\_ERROR\_RESP(  invalidSignature)  SW = 0x9000 |

Test Sequence #03 Error: Unsupported Curve

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDP\_CHALLENGE> * <S\_SMDP\_SIGNATURE1> * Set the < EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3> * Use #CERT\_S\_SM\_DPauth\_INV\_CURVE and the remaining part of the Variant A Certificates chain leading to the CI Key ID set in highest priority in the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_INV\_CURVE,  #CERT\_S\_SM\_DP\_SubCA\_SIG,  #CRL\_LIST,  TRUE  )  ) | MTD\_CHECK\_AUTH\_ERROR\_RESP(  unsupportedCurve)  SW = 0x9000 |

Test Sequence #04 Error: eUICC Challenge Mismatch

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000 |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * Change the value of <EUICC\_CHALLENGE> (retrieved in step 1) to a random value different from <EUICC\_CHALLENGE> * <S\_SMDP\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3> * Choose the #CERT\_S\_SM\_DPauth\_SIG and the remaining part of the Variant A Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  #CERT\_S\_SM\_DP\_SubCA\_SIG,  #CRL\_LIST,  TRUE  )  ) | MTD\_CHECK\_AUTH\_ERROR\_RESP(  euiccChallengeMismatch)  SW = 0x9000 |

Test Sequence #05 Error: Unknown CI PK

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDP\_CHALLENGE> * <S\_SMDP\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to a CI Key ID not present in the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3> (a random SubjectKeyIdentifier can be used) * Use #CERT\_S\_SM\_DPauth\_INV\_SIG and the remaining part of the Variant A Certificates chain leading to the CI Key ID set in highest priority in the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT (MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  #CERT\_S\_SM\_DP\_SubCA\_SIG,  #CRL\_LIST,  TRUE  )  ) | #R\_AUTH\_SERVER\_INV\_CI  SW = 0x9000  • Verify that the <S\_TRANSACTION\_ID> present in the AuthenticateResponseError is the same as <S\_TRANSACTION\_ID> in MTD\_AUTHENTICATE\_SMDP. |

Test Sequence #06 Error: Invalid Certificate Role OID

The purpose of this sequence is to make sure that the eUICC refuses any SM-DP+ Certificate for authentication that does not indicate “id-rspRole-dp-auth” in its extension for Certificate Policies.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDP\_CHALLENGE> * <S\_SMDP\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3> * Choose the #CERT\_S\_SM\_DPpb\_SIG (instead of #CERT\_S\_SM\_DPauth\_SIG) and the remaining part of the Variant A Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPpb\_SIG,  #CERT\_S\_SM\_DP\_SubCA\_SIG,  #CRL\_LIST,  TRUE  )  ) | MTD\_CHECK\_AUTH\_ERROR\_RESP(  invalidOid)  SW = 0x9000 |

Test Sequence #07 Error: No RSP session on-going

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial state |
| eUICC | No RSP session is on-going (i.e. no ES10b.getEUICCChallenge has been sent to the eUICC). |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | The following inputs are required for Step 3 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * Set <EUICC\_CHALLENGE> to a random value * <S\_SMDP\_CHALLENGE> * <S\_SMDP\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3> * Choose the #CERT\_S\_SM\_DPauth\_SIG and the remaining part of the Variant A Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  #CERT\_S\_SM\_DP\_SubCA\_SIG,  #CRL\_LIST,  TRUE  )  ) | #R\_AUTH\_SERVER\_NO\_SESSION  SW = 0x9000  The transactionId returned in the response SHALL not be checked (any value SHALL be accepted) |

Test Sequence #08 Error: With Incorrect SM-DPauth certificate – Variant A (i.e. invalid signature) – CtxParam1 with OperationType RPM

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   <S\_TRANSACTION\_ID>   <EUICC\_CHALLENGE>   <S\_SMDP\_CHALLENGE>   <S\_SMDP\_SIGNATURE1>   Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3>   Choose the #CERT\_S\_SM\_DPauth\_INV\_SIGN and the remaining part of the Variant A Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_RPM\_ICCID1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_INV\_SIGN,  #CERT\_S\_SM\_DP\_SubCA\_SIG,  #CRL\_LIST,  TRUE  )  ) | MTD\_CHECK\_AUTH\_ERROR\_RESP(  invalidCertificate)  SW = 0x9000 |

**4.2.18.2.11 TC\_eUICC\_ES10b.AuthenticateServer\_SM-DP+\_ErrorCases\_V3\_Server\_Variant\_B**

In order to execute the error cases defined in this section, the variant B of the SM-DP+ Certificates chain is used by default.

***Test Sequence #01 Error: With Incorrect SM-DPauth certificate (i.e. invalid signature)***

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:  <S\_TRANSACTION\_ID>  <EUICC\_CHALLENGE>  <S\_SMDP\_CHALLENGE>  <S\_SMDP\_SIGNATURE1>  Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3>  Choose the #CERT\_S\_SM\_DPauth\_INV\_SIGN and the remaining part of the Variant B Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_INV\_SIGN,  #CERT\_CI\_SubCA\_SIG,  #CRL\_LIST,  TRUE )  ) | MTD\_CHECK\_AUTH\_ERROR\_RESP(  invalidCertificate)  SW = 0x9000 |

***Test Sequence #02 Error: With Invalid SM-DP+ Signature***

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:  <S\_TRANSACTION\_ID>  <EUICC\_CHALLENGE>  <S\_SMDP\_CHALLENGE>  <S\_SMDP\_SIGNATURE1> NOT computed with the #SK\_S\_SM\_DPauth\_SIG but SHALL have the same length as for a valid signature  Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3>  Choose the #CERT\_S\_SM\_DPauth\_SIG and the remaining part of the Variant B Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  #CERT\_CI\_SubCA\_SIG,  #CRL\_LIST,  TRUE )  ) | MTD\_CHECK\_AUTH\_ERROR\_RESP(  invalidSignature)  SW = 0x9000 |

***Test Sequence #03 Error: Unsupported Curve***

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:  <S\_TRANSACTION\_ID>  <EUICC\_CHALLENGE>  <S\_SMDP\_CHALLENGE>  <S\_SMDP\_SIGNATURE1>  Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3>  Choose the #CERT\_S\_SM\_DPauth\_INV\_CURVE and the remaining part of the Variant B Certificates chain leading to the CI Key ID set in highest priority in the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_INV\_CURVE,  #CERT\_CI\_SubCA\_SIG,  #CRL\_LIST,  TRUE  )  ) | MTD\_CHECK\_AUTH\_ERROR\_RESP(  unsupportedCurve)  SW = 0x9000 |

***Test Sequence #04 Error: eUICC Challenge Mismatch***

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000 |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:  <S\_TRANSACTION\_ID>  Change the value of <EUICC\_CHALLENGE> (retrieved in step 1) to a random value different from <EUICC\_CHALLENGE>  <S\_SMDP\_SIGNATURE1>  Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3>  Choose the #CERT\_S\_SM\_DPauth\_SIG and the remaining part of the Variant B Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  #CERT\_CI\_SubCA\_SIG,  #CRL\_LIST,  TRUE  )  ) | MTD\_CHECK\_AUTH\_ERROR\_RESP(  euiccChallengeMismatch)  SW = 0x9000 |

***Test Sequence #05 Error: Unknown CI PK***

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDP\_CHALLENGE> * <S\_SMDP\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to a CI Key ID not present in the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3> (a random SubjectKeyIdentifier can be used) * Use #CERT\_S\_SM\_DPauth\_SIG and the remaining part of the Variant B Certificates chain leading to the CI Key ID set in highest priority in the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT (MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  #CERT\_CI\_SubCA\_SIG,  #CRL\_LIST,  TRUE  )  ) | #R\_AUTH\_SERVER\_INV\_CI  SW = 0x9000  • Verify that the <S\_TRANSACTION\_ID> present in the AuthenticateResponseError is the same as <S\_TRANSACTION\_ID> in MTD\_AUTHENTICATE\_SMDP. |

***Test Sequence #06 Error: Invalid Certificate Role OID***

The purpose of this sequence is to make sure that the eUICC refuses any SM-DP+ Certificate for authentication that does not indicate “id-rspRole-dp-auth” in its extension for Certificate Policies.

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:  <S\_TRANSACTION\_ID>  <EUICC\_CHALLENGE>  <S\_SMDP\_CHALLENGE>  <S\_SMDP\_SIGNATURE1>  Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3>  Choose the #CERT\_S\_SM\_DPpb\_SIG (instead of #CERT\_S\_SM\_DPauth\_SIG) and the remaining part of the Variant B Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPpb\_SIG,  #CERT\_CI\_SubCA\_SIG,  #CRL\_LIST,  TRUE )  ) | MTD\_CHECK\_AUTH\_ERROR\_RESP(  invalidOid)  SW = 0x9000 |

***Test Sequence #07 Error: No RSP session on-going***

|  |  |
| --- | --- |
| **Initial Conditions** |  |
| **Entity** | **Description of the initial state** |
| eUICC | No RSP session is on-going (i.e. no ES10b.getEUICCChallenge has been sent to the eUICC). |

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | The following inputs are required for Step 3 as described in the InitiateAuthentication function:  <S\_TRANSACTION\_ID>  Set <EUICC\_CHALLENGE> to a random value  <S\_SMDP\_CHALLENGE>  <S\_SMDP\_SIGNATURE1>  Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3>  Choose the #CERT\_S\_SM\_DPauth\_SIG and the remaining part of the Variant B Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  #CERT\_CI\_SubCA\_SIG,  #CRL\_LIST,  TRUE )  ) | #R\_AUTH\_SERVER\_NO\_SESSION  SW = 0x9000  The transactionId returned in the response SHALL not be checked (any value SHALL be accepted) |

***Test Sequence #08 Error: With Incorrect SM-DPauth certificate – (i.e. invalid signature) – CtxParam1 with OperationType RPM***

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   <S\_TRANSACTION\_ID>   <EUICC\_CHALLENGE>   <S\_SMDP\_CHALLENGE>   <S\_SMDP\_SIGNATURE1>   Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3>   Choose the #CERT\_S\_SM\_DPauth\_INV\_SIGN and the remaining part of the Variant B Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_RPM\_ICCID1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_INV\_SIGN,  #CERT\_CI\_SubCA\_SIG,  #CRL\_LIST,  TRUE )  ) | MTD\_CHECK\_AUTH\_ERROR\_RESP(  invalidCertificate)  SW = 0x9000 |

**4.2.18.2.12 TC\_eUICC\_ES10b.AuthenticateServer\_SM-DP+\_ErrorCases\_V3\_Server\_Variant\_C**

In order to execute the error cases defined in this section, the variant C of the SM-DP+ Certificates chain is used by default.

***Test Sequence #01 Error: With Incorrect SM-DPauth certificate (i.e. invalid signature)***

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:  <S\_TRANSACTION\_ID>  <EUICC\_CHALLENGE>  <S\_SMDP\_CHALLENGE>  <S\_SMDP\_SIGNATURE1>  Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3>  Choose the #CERT\_S\_SM\_DPauth\_INV\_SIGN and the remaining part of the Variant C Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_INV\_SIGN,  #CERT\_S\_SM\_DP\_SubCAList\_SIG,  #CRL\_LIST,  TRUE )  ) | MTD\_CHECK\_AUTH\_ERROR\_RESP(  invalidCertificate)  SW = 0x9000 |

***Test Sequence #02 Error: With Invalid SM-DP+ Signature***

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:  <S\_TRANSACTION\_ID>  <EUICC\_CHALLENGE>  <S\_SMDP\_CHALLENGE>  <S\_SMDP\_SIGNATURE1> NOT computed with the #SK\_S\_SM\_DPauth\_SIG but SHALL have the same length as for a valid signature  Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3>  Choose the #CERT\_S\_SM\_DPauth\_SIG and the remaining part of the Variant C Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  #CERT\_S\_SM\_DP\_SubCAList\_SIG,  #CRL\_LIST,  TRUE )  ) | MTD\_CHECK\_AUTH\_ERROR\_RESP(  invalidSignature)  SW = 0x9000 |

***Test Sequence #03 Error: Unsupported Curve***

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:  <S\_TRANSACTION\_ID>  <EUICC\_CHALLENGE>  <S\_SMDP\_CHALLENGE>  <S\_SMDP\_SIGNATURE1>  Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3>  Choose the #CERT\_S\_SM\_DPauth\_INV\_CURVE and the remaining part of the Variant C Certificates chain leading to the CI Key ID set in highest priority in the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_Dpauth\_INV\_CURVE,  #CERT\_S\_SM\_DP\_SubCAList\_SIG,  #CRL\_LIST,  TRUE  )  ) | MTD\_CHECK\_AUTH\_ERROR\_RESP(  unsupportedCurve)  SW = 0x9000 |

***Test Sequence #04 Error: eUICC Challenge Mismatch***

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000 |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:  <S\_TRANSACTION\_ID>  Change the value of <EUICC\_CHALLENGE> (retrieved in step 1) to a random value different from <EUICC\_CHALLENGE>  <S\_SMDP\_SIGNATURE1>  Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3>  Choose the #CERT\_S\_SM\_DPauth\_SIG and the remaining part of the Variant C Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_Dpauth\_SIG,  #CERT\_S\_SM\_DP\_SubCAList\_SIG,  #CRL\_LIST,  TRUE  )  ) | MTD\_CHECK\_AUTH\_ERROR\_RESP(  euiccChallengeMismatch)  SW = 0x9000 |

***Test Sequence #05 Error: Unknown CI PK***

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDP\_CHALLENGE> * <S\_SMDP\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to a CI Key ID not present in the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3> (a random SubjectKeyIdentifier can be used) * Use #CERT\_S\_SM\_DPauth\_SIG and the remaining part of the Variant C Certificates chain leading to the CI Key ID set in highest priority in the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT (MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  #CERT\_S\_SM\_DP\_SubCAList\_SIG,  #CRL\_LIST,  TRUE  )  ) | #R\_AUTH\_SERVER\_INV\_CI  SW = 0x9000  • Verify that the <S\_TRANSACTION\_ID> present in the AuthenticateResponseError is the same as <S\_TRANSACTION\_ID> in MTD\_AUTHENTICATE\_SMDP. |

***Test Sequence #06 Error: Invalid Certificate Role OID***

The purpose of this sequence is to make sure that the eUICC refuses any SM-DP+ Certificate for authentication that does not indicate “id-rspRole-dp-auth” in its extension for Certificate Policies.

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:  <S\_TRANSACTION\_ID>  <EUICC\_CHALLENGE>  <S\_SMDP\_CHALLENGE>  <S\_SMDP\_SIGNATURE1>  Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3>  Choose the #CERT\_S\_SM\_DPpb\_SIG (instead of #CERT\_S\_SM\_DPauth\_SIG) and the remaining part of the Variant C Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPpb\_SIG,  #CERT\_S\_SM\_DP\_SubCAList\_SIG,  #CRL\_LIST,  TRUE )  ) | MTD\_CHECK\_AUTH\_ERROR\_RESP(  invalidOid)  SW = 0x9000 |

***Test Sequence #07 Error: No RSP session on-going***

|  |  |
| --- | --- |
| **Initial Conditions** |  |
| **Entity** | **Description of the initial state** |
| eUICC | No RSP session is on-going (i.e. no ES10b.getEUICCChallenge has been sent to the eUICC). |

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | The following inputs are required for Step 3 as described in the InitiateAuthentication function:  <S\_TRANSACTION\_ID>  Set <EUICC\_CHALLENGE> to a random value  <S\_SMDP\_CHALLENGE>  <S\_SMDP\_SIGNATURE1>  Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3>  Choose the #CERT\_S\_SM\_DPauth\_SIG and the remaining part of the Variant C Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  #CERT\_S\_SM\_DP\_SubCAList\_SIG,  #CRL\_LIST,  TRUE )  ) | #R\_AUTH\_SERVER\_NO\_SESSION  SW = 0x9000  The transactionId returned in the response SHALL not be checked (any value SHALL be accepted) |

***Test Sequence #08 Error: With Incorrect SM-DPauth certificate – (i.e. invalid signature) – CtxParam1 with OperationType RPM***

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   <S\_TRANSACTION\_ID>   <EUICC\_CHALLENGE>   <S\_SMDP\_CHALLENGE>   <S\_SMDP\_SIGNATURE1>   Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3>   Choose the #CERT\_S\_SM\_DPauth\_INV\_SIGN and the remaining part of the Variant C Certificates chain leading to the same Root CI certificate as the one chosen for signing | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_RPM\_ICCID1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_INV\_SIGN,  #CERT\_S\_SM\_DP\_SubCAList\_SIG,  #CRL\_LIST,  TRUE )  ) | MTD\_CHECK\_AUTH\_ERROR\_RESP(  invalidCertificate)  SW = 0x9000 |

**4.2.18.2.13 TC\_eUICC\_ES10b.AuthenticateServerV3\_SM-DP+\_Independently\_Of\_Server\_Variant**

The purpose of the test sequences in this Test Case is to verify that regardless of the Variant of eUICC Certificate chain configured in the eUICC, the eUICC is able to verify the SM-DP+ Certificates chain, including if the server uses a different Variant.

This Test Case does not aim to test all combinations of options, so a single one is chosen, using MatchingId and no IMEI in CtxParams1.

Test Sequence #01 Nominal: with server Certificate chain Variant O

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify that it contains at least #IUT\_SERVER\_CI\_PK\_ID  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING> and < EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3> from response data and verify that at least one of them contains #IUT\_EUICC\_CERT\_CHAIN\_VARIANT \_CI\_PK\_ID |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDP\_CHALLENGE> * <S\_SMDP\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> and <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to #IUT\_EUICC\_CERT\_CHAIN\_VARIANT \_CI\_PK\_ID * Choose the #CERT\_S\_SM\_DPauth\_SIG leading to the Root CI identified by #IUT\_SERVER\_CI\_PK\_ID | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(MTD\_AUTHENTICATE\_SMDP\_USING\_IUT\_VARIANT (  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_MATCH\_ID,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  NO\_PARAM,  #CRL\_LIST  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_MATCH\_ID  )  SW = 0x9000 |

Test Sequence #02 Nominal: with server Certificate chain Variant A

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify that it contains at least #IUT\_SERVER\_CI\_PK\_ID  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING> and < EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3> from response data and verify that at least one of them contains #IUT\_EUICC\_CERT\_CHAIN\_VARIANT \_CI\_PK\_ID |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDP\_CHALLENGE> * <S\_SMDP\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> and <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to #IUT\_EUICC\_CERT\_CHAIN\_VARIANT \_CI\_PK\_ID * Choose the #CERT\_S\_SM\_DPauth\_SIG and the remaining part of the Variant A Certificates chain leading to the Root CI identified by #IUT\_SERVER\_CI\_PK\_ID | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP\_USING\_IUT\_VARIANT (  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_MATCH\_ID,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  #CERT\_S\_SM\_DP\_SubCA\_SIG,  #CRL\_LIST  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_MATCH\_ID)  SW = 0x9000 |

***Test Sequence #03 Nominal: with Server Certificate chain Variant B***

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify that it contains at least #IUT\_SERVER\_CI\_PK\_ID  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING> and < EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3> from response data and verify that at least one of them contains #IUT\_EUICC\_CERT\_CHAIN\_VARIANT \_CI\_PK\_ID |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDP\_CHALLENGE> * <S\_SMDP\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> and <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to #IUT\_EUICC\_CERT\_CHAIN\_VARIANT \_CI\_PK\_ID * Choose the #CERT\_S\_SM\_DPauth\_SIG and the remaining part of the Variant B Certificates chain leading to the Root CI identified by #IUT\_SERVER\_CI\_PK\_ID | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP\_USING\_IUT\_VARIANT (  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_MATCH\_ID,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  #CERT\_CI\_SubCA\_SIG,  #CRL\_LIST  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_MATCH\_ID)  SW = 0x9000 |

***Test Sequence #04 Nominal: with Server Certificate chain Variant C***

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Direction** | **Sequence / Description** | **Expected result** |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify that it contains at least #IUT\_SERVER\_CI\_PK\_ID  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING> and < EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3> from response data and verify that at least one of them contains #IUT\_EUICC\_CERT\_CHAIN\_VARIANT \_CI\_PK\_ID |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDP\_CHALLENGE> * <S\_SMDP\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> and <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to #IUT\_EUICC\_CERT\_CHAIN\_VARIANT \_CI\_PK\_ID * Choose the #CERT\_S\_SM\_DPauth\_SIG and the remaining part of the Variant C Certificates chain leading to the Root CI identified by #IUT\_SERVER\_CI\_PK\_ID | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP\_USING\_IUT\_VARIANT (  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_MATCH\_ID,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  #CERT\_S\_SM\_DP\_SubCAList\_SIG,  #CRL\_LIST  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1\_MATCH\_ID)  SW = 0x9000 |

**4.2.18.2.14 TC\_eUICC\_ES10b.AuthenticateServerV3\_SM-DS\_Independently\_Of\_Server\_Variant**

The purpose of the test sequences in this Test Case is to verify that regardless of the Variant of eUICC Certificate chain configured in the eUICC, the eUICC is able to verify the SM-DS Certificates chain, including if the server uses a different Variant.

This Test Case does not aim to test all combinations of options, so a single one is chosen, using EventId and no IMEI in CtxParams1.

Test Sequence #01 Nominal: with server Certificate Chain Variant O

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify that it contains at least #IUT\_SERVER\_CI\_PK\_ID  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING> and < EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3> from response data and verify that at least one of them contains #IUT\_EUICC\_CERT\_CHAIN\_VARIANT \_CI\_PK\_ID |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| 3 | The following inputs are required for Step 4 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDS\_CHALLENGE> * <S\_SMDS\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> and <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3> to #IUT\_EUICC\_CERT\_CHAIN\_VARIANT \_CI\_PK\_ID * Choose the #CERT\_S\_SM\_DPauth\_SIG leading to the Root CI identified by #IUT\_SERVER\_CI\_PK\_ID | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDS\_USING\_IUT\_VARIANT (  #TEST\_ROOT\_DS\_ADDRESS,  <S\_SMDS\_CHALLENGE>,  #CTX\_PARAMS1\_EVENT\_ID,  <S\_SMDS\_SIGNATURE1>,  #CERT\_S\_SM\_DSauth\_SIG,  NO\_PARAM,  #CRL\_LIST  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_ROOT\_DS\_ADDRESS,  <S\_SMDS\_CHALLENGE>,  #CTX\_PARAMS1\_EVENT\_ID)  SW = 0x9000 |

Test Sequence #02 Nominal: with server Certificate Chain Variant A

This test sequence is FFS.

Test Sequence #03 Nominal: with server Certificate Chain Variant B

This test sequence is FFS.

Test Sequence #04 Nominal: with server Certificate Chain Variant C

This test sequence is FFS.

### 4.2.19 ES10b (LPA -- eUICC): CancelSession

#### 4.2.19.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 2.6.7.2
* Section 3.0.2, 3.1.3, 3.1.3.2, 3.1.3.3
* Section 5.7.14

#### 4.2.19.2 Test Cases

##### 4.2.19.2.1 TC\_eUICC\_ES10b.CancelSessionNIST

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is not loaded on the eUICC. |
| eUICC | The communication between the S\_Device and the eUICC has been initialized and the S\_LPAd has selected the ISD-R.  Common Mutual Authentication procedure has been successfully executed between the eUICC and the S\_SM-DP+ (i.e. the response has been sent by the eUICC for ES10b.AuthenticateServer)   * the same GSMA CI based on NIST P-256 curve has been chosen for signing and for verification |

Test Sequence #01 Nominal: End User Rejection

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #CANCEL\_SESSION\_REJECT) | #R\_CANCEL\_SESSION\_REJ  SW = 0x9000  The <EUICC\_CS\_SIGNATURE> SHALL be verified with the #PK\_EUICC\_SIG.  Verify that the <S\_TRANSACTION\_ID> present in the response is the same as in #CANCEL\_SESSION\_REJECT |
| 2 | PROC\_VERIFY\_SESSION\_IS\_CANCELLED | | |

Test Sequence #02 Nominal: End User Postponed

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #CANCEL\_SESSION\_POSTPONED) | #R\_CANCEL\_SESSION\_POSTPONED  SW = 0x9000  The <EUICC\_CS\_SIGNATURE> SHALL be verified with the #PK\_EUICC\_SIG.  Verify that the <S\_TRANSACTION\_ID> present in the response is the same as in #CANCEL\_SESSION\_POSTPONED |
| 2 | PROC\_VERIFY\_SESSION\_IS\_CANCELLED | | |

Test Sequence #03 Nominal: Timeout

The RSP session is delayed because the End User does not confirm the download of the Profile within the timeout interval defined by the LPAd.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #CANCEL\_SESSION\_TIMEOUT) | #R\_CANCEL\_SESSION\_TIMEOUT  SW = 0x9000  The <EUICC\_CS\_SIGNATURE> SHALL be verified with the #PK\_EUICC\_SIG.  Verify that the <S\_TRANSACTION\_ID> present in the response is the same as in #CANCEL\_SESSION\_TIMEOUT |
| 2 | PROC\_VERIFY\_SESSION\_IS\_CANCELLED | | |

Test Sequence #04 Nominal: PPR not allowed

The RSP session is terminated because the LPAd detected that PPR(s) set in the Profile Metadata is/are not allowed.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #CANCEL\_SESSION\_PPR) | #R\_CANCEL\_SESSION\_PPR  SW = 0x9000  The <EUICC\_CS\_SIGNATURE> SHALL be verified with the #PK\_EUICC\_SIG.  Verify that the <S\_TRANSACTION\_ID> present in the response is the same as in #CANCEL\_SESSION\_PPR |
| 2 | PROC\_VERIFY\_SESSION\_IS\_CANCELLED | | |

Test Sequence #05 Nominal: Metadata Mismatch

The RSP session is terminated because the LPAd detected that the Profile Metadata in the response to "ES9+.AuthenticateClient" does not match the Profile Metadata in the Bound Profile Package.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | Sub-procedure Profile Download and Installation – End User Confirmation has been successfully executed between the eUICC and the S\_SM-DP+ (i.e. the response has been sent by the eUICC for ES10b.PrepareDownload)   * #PREP\_DOWNLOAD\_NO\_CC has been sent to the eUICC |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #CANCEL\_SESSION\_METADATA) | #R\_CANCEL\_SESSION\_METADATA  SW = 0x9000  The <EUICC\_CS\_SIGNATURE> SHALL be verified with the #PK\_EUICC\_SIG.  Verify that the <S\_TRANSACTION\_ID> present in the response is the same as in #CANCEL\_SESSION\_METADATA |
| 2 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| 3 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_OP\_PROF1,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| 4 | Split the <BPP> into several segments arrays named:   <BPP\_SEG\_INIT>   <BPP\_SEG\_A0>   <BPP\_SEG\_A1>   <BPP\_SEG\_A3> | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x6985 or 0x6A88 |
| 6 | PROC\_VERIFY\_SESSION\_IS\_CANCELLED | | |

Test Sequence #06 Nominal: BPP Parsing Error

The RSP session is terminated because the LPAd has encountered an error while parsing the Bound Profile Package received from the SM-DP+.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | Sub-procedure Profile Download and Installation – End User Confirmation has been successfully executed between the eUICC and the S\_SM-DP+ (i.e. the response has been sent by the eUICC for ES10b.PrepareDownload)   * #PREP\_DOWNLOAD\_NO\_CC has been sent to the eUICC |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #CANCEL\_SESSION\_LOAD\_BPP) | #R\_CANCEL\_SESSION\_LOAD\_BPP  SW = 0x9000  The <EUICC\_CS\_SIGNATURE> SHALL be verified with the #PK\_EUICC\_SIG.  Verify that the <S\_TRANSACTION\_ID> present in the response is the same as in #CANCEL\_SESSION\_LOAD\_BPP |
| 2 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| 3 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_OP\_PROF1,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| 4 | Split the <BPP> into several segments arrays named:   <BPP\_SEG\_INIT>   <BPP\_SEG\_A0>   <BPP\_SEG\_A1>   <BPP\_SEG\_A3> | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x6985 or 0x6A88 |
| 6 | PROC\_VERIFY\_SESSION\_IS\_CANCELLED | | |

Test Sequence #07 Nominal: Load BPP Execution Error

The RSP session is terminated because the LPAd has encountered an error while installing the Bound Profile Package received from the SM-DP+.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | Sub-procedure Profile Download and Installation – End User Confirmation has been successfully executed between the eUICC and the S\_SM-DP+ (i.e. the response has been sent by the eUICC for ES10b.PrepareDownload)   * #PREP\_DOWNLOAD\_NO\_CC has been sent to the eUICC |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| IC2 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_PROF1,  #METADATA\_OP\_PROF1,  NO\_PARAM,  #UPP\_OP\_PROF1) | | |
| IC3 | Split the <BPP> into several segments arrays named:   <BPP\_SEG\_INIT>   <BPP\_SEG\_A0>   <BPP\_SEG\_A1>   <BPP\_SEG\_A3> | | |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #CANCEL\_SESSION\_LOAD\_BPP) | #R\_CANCEL\_SESSION\_LOAD\_BPP  SW = 0x9000  The <EUICC\_CS\_SIGNATURE> SHALL be verified with the #PK\_EUICC\_SIG.  Verify that the <S\_TRANSACTION\_ID> present in the response is the same as in #CANCEL\_SESSION\_LOAD\_BPP |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0) | SW=0x6985 or 0x6A88 |
| 3 | PROC\_VERIFY\_SESSION\_IS\_CANCELLED | | |

Test Sequence #08 Nominal: Undefined Reason

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #CANCEL\_SESSION\_UNDEF) | #R\_CANCEL\_SESSION\_UNDEF  SW = 0x9000  The <EUICC\_CS\_SIGNATURE> SHALL be verified with the #PK\_EUICC\_SIG.  Verify that the <S\_TRANSACTION\_ID> present in the response is the same as in #CANCEL\_SESSION\_UNDEF |
| 2 | PROC\_VERIFY\_SESSION\_IS\_CANCELLED | | |

##### 4.2.19.2.2 TC\_eUICC\_ES10b.CancelSessionBRP

In these test sequences, once the RSP session has been cancelled, verifications are performed in order to check that it is neither possible to execute the Download Confirmation procedure nor to execute the Common Mutual Authentication procedure by referring to the cancelled TransactionID.

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is not loaded on the eUICC. |
| eUICC | The communication between the S\_Device and the eUICC has been initialized and the S\_LPAd has selected the ISD-R.  Common Mutual Authentication procedure has been successfully executed between the eUICC and the S\_SM-DP+   * the same GSMA CI based on BrainpoolP256r1 curve has been chosen for signing and for verification |

Test Sequence #01 Nominal: End User Rejection

This test sequence SHALL be the same as the Test Sequence #01 defined in section 4.2.19.2.1 – TC\_eUICC\_ES10b.CancelSessionNIST except that all keys and certificates SHALL be based on BrainpoolP256r1.

Test Sequence #02 Nominal: End User Postponed

This test sequence SHALL be the same as the Test Sequence #02 defined in section 4.2.19.2.1 – TC\_eUICC\_ES10b.CancelSessionNIST except that all keys and certificates SHALL be based on BrainpoolP256r1.

Test Sequence #03 Nominal: Timeout

This test sequence SHALL be the same as the Test Sequence #03 defined in section 4.2.19.2.1 – TC\_eUICC\_ES10b.CancelSessionNIST except that all keys and certificates SHALL be based on BrainpoolP256r1.

Test Sequence #04 Nominal: PPR not allowed

This test sequence SHALL be the same as the Test Sequence #04 defined in section 4.2.19.2.1 – TC\_eUICC\_ES10b.CancelSessionNIST except that all keys and certificates SHALL be based on BrainpoolP256r1.

Test Sequence #05 Nominal: Metadata Mismatch

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial state |
| eUICC | Sub-procedure Profile Download and Installation – End User Confirmation has been successfully executed between the eUICC and the S\_SM-DP+   * #PREP\_DOWNLOAD\_NO\_CC has been sent to the eUICC |

This test sequence SHALL be the same as the Test Sequence #05 defined in section 4.2.19.2.1 – TC\_eUICC\_ES10b.CancelSessionNIST except that all keys and certificates SHALL be based on BrainpoolP256r1.

Test Sequence #06 Nominal: BPP Parsing Error

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial state |
| eUICC | Sub-procedure Profile Download and Installation – End User Confirmation has been successfully executed between the eUICC and the S\_SM-DP+   * #PREP\_DOWNLOAD\_NO\_CC has been sent to the eUICC |

This test sequence SHALL be the same as the Test Sequence #06 defined in section 4.2.19.2.1 – TC\_eUICC\_ES10b.CancelSessionNIST except that all keys and certificates SHALL be based on BrainpoolP256r1.

Test Sequence #07 Nominal: Load BPP Execution Error

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial state |
| eUICC | Sub-procedure Profile Download and Installation – End User Confirmation has been successfully executed between the eUICC and the S\_SM-DP+   * #PREP\_DOWNLOAD\_NO\_CC has been sent to the eUICC |

This test sequence SHALL be the same as the Test Sequence #07 defined in section 4.2.19.2.1 – TC\_eUICC\_ES10b.CancelSessionNIST except that all keys and certificates SHALL be based on BrainpoolP256r1.

Test Sequence #08 Nominal: Undefined Reason

This test sequence SHALL be the same as the Test Sequence #08 defined in section 4.2.19.2.1 – TC\_eUICC\_ES10b.CancelSessionNIST except that all keys and certificates SHALL be based on BrainpoolP256r1.

##### 4.2.19.2.3 VOID

##### 4.2.19.2.4 TC\_eUICC\_ES10b.CancelSession\_ErrorCase

Test Sequence #01 Error: No on-going RSP session

On receiving a CancelSession request whereas there is no on-going RSP session, the eUICC SHALL return an error code.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | No RSP session is on-going (i.e. no Common Mutual Authentication procedure has been executed). |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #CANCEL\_SESSION\_INV\_TRANS\_ID) | #R\_CANCEL\_SESSION\_INV\_TRANS\_ID  SW = 0x9000 |

Test Sequence #02 Error: Invalid Transaction ID

On receiving a CancelSession request with a TransactionID different from the on-going one, the eUICC SHALL not discard the current RSP session and return an error code.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The communication between the S\_Device and the eUICC has been initialized and the S\_LPAd has selected the ISD-R.   * #GET\_EUICC\_INFO1, #GET\_EUICC\_CHALLENGE and MTD\_AUTHENTICATE\_SMDP have been sent to the eUICC as defined in section 2.2.3.5 |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #CANCEL\_SESSION\_INV\_TRANS\_ID) | #R\_CANCEL\_SESSION\_INV\_TRANS\_ID  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  #PREP\_DOWNLOAD\_NO\_CC) | #R\_PREP\_DOWNLOAD\_NO\_CC  SW=0x9000 |

### 4.2.20 ES10c (LPA -- eUICC): GetProfilesInfo

#### 4.2.20.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 2.4.5.2
* Section 3.1.4
* Section 3.2.4
* Section 5.7.15

#### 4.2.20.2 Test Cases

##### 4.2.20.2.1 TC\_eUICC\_ES10c.GetProfilesInfo

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |
| eUICC | The Nickname of PROFILE\_OPERATIONAL1 and PROFILE\_OPERATIONAL2 is empty. |
| eUICC | The Nickname of the PROFILE\_OPERATIONAL3 is equal to #NICKNAME3. |

Test Sequence #01 Nominal: Get All Profiles

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk: {  #PROFILE\_INFO1,  #PROFILE\_INFO2,  #PROFILE\_INFO3  }  SW = 0x9000 |

Test Sequence #02 Nominal: Get Profile by ICCID

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW = 0x9000 |

Test Sequence #03 Nominal: Get Profile by AID

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW = 0x9000 |

Test Sequence #04 Nominal: Get All Operational Profiles

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_PROFILES\_INFO\_PROFCLASS) | response ProfileInfoListResponse::= profileInfoListOk: {  #PROFILE\_INFO1,  #PROFILE\_INFO2,  #PROFILE\_INFO3  }  SW = 0x9000 |

Test Sequence #05 Nominal: Get Profile ICCID list

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_PROFILES\_INFO\_TAGLIST\_ICCID) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILES\_INFO\_TAGLIST\_ICCID  }  SW = 0x9000 |

Test Sequence #06 Nominal: Get Profile AID list

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_PROFILES\_INFO\_TAGLIST\_ISDPAID) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILES\_INFO\_TAGLIST\_ISDPAID  }  SW = 0x9000 |

Test Sequence #07 Nominal: Get Profile Nickname list

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_PROFILES\_INFO\_TAGLIST\_PROFILE\_NICKNAME) | response ProfileInfoListResponse::= profileInfoListOk : {  …  #PROFILES\_INFO\_TAGLIST\_PROFILE\_NICKNAME  ...  }  SW = 0x9000 |

Test Sequence #08 Nominal: Get Profile SP Name list

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_PROFILES\_INFO\_TAGLIST\_SP\_NAME) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILES\_INFO\_TAGLIST\_SP\_NAME  }  SW = 0x9000 |

Test Sequence #09 Nominal: Get Profile Name list

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_PROFILES\_INFO\_TAGLIST\_PROFILE\_NAME) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILES\_INFO\_TAGLIST\_PROFILE\_NAME  }  SW = 0x9000 |

Test Sequence #10 Nominal: Get Profile Icon list

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step | Direction | | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_PROFILES\_INFO\_TAGLIST\_ICON) | | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILES\_INFO\_TAGLIST\_ICON  }  SW = 0x9000 |

Test Sequence #11 Nominal: Get Profile Owner list

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_PROFILES\_INFO\_TAGLIST\_PROFILE\_OWNER) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILES\_INFO\_TAGLIST\_PROFILE\_OWNER  }  SW = 0x9000 |

Test Sequence #12 Nominal: Get Profile SM-DP+ proprietary data list

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC with dpProprietaryData #SMDP\_PROP\_DATA1 (i.e. #CONF\_ISDP\_PROF1 is used during the Profile downloading). |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_PROFILES\_INFO\_TAGLIST\_SMDP\_PROP\_DATA) | response ProfileInfoListResponse::= profileInfoListOk : {  …  #PROFILES\_INFO\_TAGLIST\_SMDP\_PROP\_DATA  …  }  SW = 0x9000 |

Test Sequence #13 Nominal: Get Profile ICCID and State list

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_PROFILES\_INFO\_TAGLIST1) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILES\_INFO\_TAGLIST1  }  SW = 0x9000 |

Test Sequence #14 Nominal: Get Profile Nickname and State list

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_PROFILES\_INFO\_TAGLIST2) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILES\_INFO\_TAGLIST2  }  SW = 0x9000 |

Test Sequence #15 Nominal: Get Profile Icon and Icon Type list

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_PROFILES\_INFO\_TAGLIST3) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILES\_INFO\_TAGLIST3  }  SW = 0x9000 |

Test Sequence #16 Nominal: Get Profile Icon and State list

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_PROFILES\_INFO\_TAGLIST4) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILES\_INFO\_TAGLIST4  }  SW = 0x9000 |

Test Sequence #17 Nominal: Get Operational Profile ICCID and State list

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_PROFILES\_INFO\_OPTAGLIST1) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILES\_INFO\_TAGLIST1  }  SW = 0x9000 |

Test Sequence #18 Nominal: Get Operational Profile Nickname and State list

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_PROFILES\_INFO\_OPTAGLIST2) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILES\_INFO\_TAGLIST2  }  SW = 0x9000 |

Test Sequence #19 Nominal: Get Operational Profile Icon and Icon type list

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_PROFILES\_INFO\_OPTAGLIST3) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILES\_INFO\_TAGLIST3  }  SW = 0x9000 |

Test Sequence #20 Nominal: Get Operational Profile Icon and State list

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_PROFILES\_INFO\_OPTAGLIST4) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILES\_INFO\_TAGLIST4  }  SW = 0x9000 |

Test Sequence #21 Nominal: Get Profile State of the defined Profile

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_PROFILES\_INFO\_ICCID\_TAGLIST1) | response ProfileInfoListResponse::= profileInfoListOk : { #PROFILES\_INFO\_ICCID\_TAGLIST1  }  SW = 0x9000 |

Test Sequence #22 Nominal: Get Profile Icon Type of the defined Profile

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_PROFILES\_INFO\_ICCID\_TAGLIST2) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILES\_INFO\_ICCID\_TAGLIST2  }  SW = 0x9000 |

Test Sequence #23 Nominal: Get Profile Class of the defined Profile

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_PROFILES\_INFO\_ICCID\_TAGLIST3) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILES\_INFO\_ICCID\_TAGLIST3  }  SW = 0x9000 |

Test Sequence #24 Nominal: Get Notification Configuration of the defined Profile

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_PROFILES\_INFO\_ICCID\_TAGLIST4) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILES\_INFO\_ICCID\_TAGLIST4  }  SW = 0x9000 |

Test Sequence #25 Nominal: Get Profile Policy Rules of the defined Profile

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_PROFILES\_INFO\_ICCID\_TAGLIST5) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILES\_INFO\_ICCID\_TAGLIST5  }  SW = 0x9000 |

Test Sequence #26 Nominal: Get empty Profile list. No Profile installed

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | No Profile is loaded on the eUICC (this condition overrides the general initial condition defined in this test case). |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (#GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk: {  }  SW = 0x9000 |

##### 4.2.20.2.2 TC\_eUICC\_ES10c.GetProfilesInfo\_ErrorCases

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |
| eUICC | The Nickname of PROFILE\_OPERATIONAL1 is empty. |

Test Sequence #01 Error: Get Profiles during a Profile Enabling

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | No response data is returned  SW=0x91XX |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | SW = 0x6985 |
| 3 | S\_Device → eUICC | FETCH 'XX' | REFRESH Command (“UICC Reset”) |
| 4 | Repeat IC1 and IC2 | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk: {  #PROFILE\_INFO1  }  SW = 0x9000 |
| 6 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 7 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

Test Sequence #02 Error: Get Profiles during a Profile Disabling

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | No response data is returned  SW=0x91XX |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | SW = 0x6985 |
| 3 | S\_Device → eUICC | FETCH 'XX' | REFRESH Command (“UICC Reset”) |
| 4 | Repeat IC1 and IC2 | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk: {  #PROFILE\_INFO1\_DISABLED  }  SW = 0x9000 |
| 6 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #03 Error: Get Profiles during an eUICC Memory Reset

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x91XX |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | SW = 0x6985 |
| 3 | S\_Device → eUICC | FETCH 'XX' | REFRESH Command (“UICC Reset”) |
| 4 | Repeat IC1 and IC2 | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk: {  }  SW = 0x9000 |

##### 4.2.20.2.3 TC\_eUICC\_ES10c.GetProfilesInfo\_MEPA1

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |
| eUICC | The Nickname of PROFILE\_OPERATIONAL1 and PROFILE\_OPERATIONAL2 is empty. |
| eUICC | The Nickname of the PROFILE\_OPERATIONAL3 is equal to #NICKNAME3. |

Test Sequence #01 Nominal: Get Profile ICCID, State and enable on eSIM Port list

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | eUICC is MEP capable |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on Port 1 |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on Port 2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO\_TAGLIST (  ‘5A9F709F24’)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILES\_INFO\_TAGLIST5\_MEPA1;  }  SW=0x9000 |

##### 4.2.20.2.4 TC\_eUICC\_ES10c.GetProfilesInfo\_MEPA2

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |
| eUICC | The Nickname of PROFILE\_OPERATIONAL1 and PROFILE\_OPERATIONAL2 is empty. |
| eUICC | The Nickname of the PROFILE\_OPERATIONAL3 is equal to #NICKNAME3. |

Test Sequence #01 Nominal: Get Profile ICCID, State and enable on eSIM Port list

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | eUICC is MEP capable |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO\_TAGLIST (  ‘5A9F709F24’)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILES\_INFO\_TAGLIST5\_MEPA2\_12  OR  PROFILES\_INFO\_TAGLIST5\_MEPA2\_21;  }  SW=0x9000 |

##### 4.2.20.2.5 TC\_eUICC\_ES10c.GetProfilesInfo\_MEPB

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |
| eUICC | The Nickname of PROFILE\_OPERATIONAL1 and PROFILE\_OPERATIONAL2 is empty. |
| eUICC | The Nickname of the PROFILE\_OPERATIONAL3 is equal to #NICKNAME3. |

Test Sequence #01 Nominal: Get Profile ICCID, State and enable on eSIM Port list

The purpose of this test sequence is to verify that the fields profileState and enabledOnEsimPort properly reflect the Profile’s state depending on the Command Port.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | eUICC is MEP capable |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on Port 0 |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on Port 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS  Verify  <MEP\_MAX\_LSIS> = #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO\_TAGLIST\_MEP (  ‘5A9F709F24’)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILES\_INFO\_TAGLIST5\_MEPB  }  SW=0x9000  Verify that:   * the ProfileInfo object with #ICCID\_OP\_PROF1 has its profileState equal to enabled and its enabledOnEsimPort equal to 0 * the ProfileInfo object with #ICCID\_OP\_PROF2 has its enabledOnEsimPort equal to 1 |
| 2 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 3 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO\_TAGLIST\_MEP (  ‘5A9F709F24’)) | Verify that:   * the ProfileInfo object with #ICCID\_OP\_PROF2 has its profileState equal to enabled and its enabledOnEsimPort equal to 1 * the ProfileInfo object with #ICCID\_OP\_PROF1 has its enabledOnEsimPort equal to 0 |

### 4.2.21 ES10c (LPA -- eUICC): EnableProfile

#### 4.2.21.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 2.4.3
* Section 2.9.1, 2.9.3.3
* Section 3.2.1
* Section 3.4.3
* Section 5.7.16

#### 4.2.21.2 Test Cases

##### 4.2.21.2.1 TC\_eUICC\_ES10c.EnableProfile\_Case3

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |

Test Sequence #01 Nominal: Enable Profile by ISD-P AID and “refreshFlag” set when Device supports “UICC Reset”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | No response data is returned  SW=0x91XX |
| 2 | S\_Device → eUICC | FETCH 'XX' | REFRESH Command (“UICC Reset”) |
| 3 | Repeat IC1 and IC2 | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 5 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 6 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

Test Sequence #02 Nominal: Enable Profile by ICCID and “refreshFlag” set when Device supports “UICC Reset”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | No response data is returned  SW=0x91XX |
| 2 | S\_Device → eUICC | FETCH 'XX' | REFRESH Command (“UICC Reset”) |
| 3 | Repeat IC1 and IC2 | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 5 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 6 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

Test Sequence #03 Nominal: Enable Profile by ISD-P AID and “refreshFlag” set when Device supports “eUICC Profile State Change”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | No response data is returned  SW=0x91XX |
| 2 | S\_Device → eUICC | FETCH 'XX' | REFRESH Command (“eUICC Profile State change”) |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | Execute IC1 from step 2 to step 4 | | |
| 5 | Repeat IC2 | | |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 7 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 8 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

Test Sequence #04 Nominal: Enable Profile by ICCID and “refreshFlag” set when Device supports “eUICC Profile State Change”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | No response data is returned  SW=0x91XX |
| 2 | S\_Device → eUICC | FETCH 'XX' | REFRESH Command (“eUICC Profile State change”) |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | Execute IC1 from step 2 to step 4 | | |
| 5 | Repeat IC2 | | |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 7 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 8 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

Test Sequence #05 Nominal: Enable Profile by ISD-P AID and “refreshFlag” not set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  FALSE)) | No response data is returned  SW=0x9000 |
| 2 | S\_Device → eUICC | [TERMINAL\_PROFILE] | Toolkit initialization THEN SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 4 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 5 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

Test Sequence #06 Nominal: Enable Profile by ICCID and “refreshFlag” not set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  FALSE)) | No response data is returned  SW=0x9000 |
| 2 | S\_Device → eUICC | [TERMINAL\_PROFILE] | Toolkit initialization THEN SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 4 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 5 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

Test Sequence #07 Nominal: Enable Profile by ICCID with refreshFLag set while proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x9100 |
| IC4 | Do not send FETCH command | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF2,  NO\_PARAM,  TRUE)) | No response data is returned  SW=0x91YY |
| 2 | S\_Device 🡪eUICC | FETCH 'YY' | REFRESH Command (“UICC Reset”) |
| 3 | Repeat IC1 and IC2 | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED,  #PROFILE\_INFO2\_DISABLED  }  SW=0x9000 |
| 5 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 6 | S\_Device 🡪 eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF2  SW=0x9000 |

Test Sequence #08 Nominal: Enable Profile by ICCID with refreshFLag not set while proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x9100 |
| IC4 | Do not send FETCH command | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF2,  NO\_PARAM,  FALSE)) | No response data is returned  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED,  #PROFILE\_INFO2\_ENABLED  }  SW=0x9000 |
| 3 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 4 | S\_Device 🡪 eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF2  SW=0x9000 |

Test Sequence #09 Nominal: Enable Profile by ICCID with refreshFLag set while proactive session is ongoing with Terminal Response outstanding – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| 1 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF2,  NO\_PARAM,  TRUE)) | No response data returned SW=0x9000 |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x91YY |
| 4 | S\_Device 🡪eUICC | FETCH 'YY' | REFRESH Command (“UICC Reset”) |
| 5 | Repeat IC1 and IC2 | | |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED,  #PROFILE\_INFO2\_ENABLED  }  SW=0x9000 |
| 7 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 8 | S\_Device 🡪 eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF2  SW=0x9000 |

Test Sequence #10 Nominal: Enable Profile by ICCID with refreshFLag not set while proactive session is ongoing with Terminal Response outstanding – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| 1 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF2,  NO\_PARAM,  FALSE)) | No response data is returned  SW=0x9000 |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=any value except 91XX |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED,  #PROFILE\_INFO2\_ENABLED  }  SW=0x9000 |
| 5 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 6 | S\_Device 🡪 eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF2  SW=0x9000 |

##### 4.2.21.2.2 TC\_eUICC\_ES10c.EnableProfile\_ErrorCases\_Case3

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |

Test Sequence #01 Error: Enable Profile by an unknown ISD-P AID

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The Operational Profile identified by the ISD-P AID <ISD\_P\_AIDX> is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AIDX>,  TRUE)) | SW=0x6A82 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #02 Error: Enable Profile by an unknown ICCID

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The Operational Profile identified by the ICCID #ICCID\_OP\_PROFX is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROFX,  NO\_PARAM,  TRUE)) | SW=0x6A82 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #03 Error: Enable Profile (by ISD-P AID) is not possible when this Operational Profile is in Enabled state

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | SW=0x6985 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #04 Error: Enable Profile (by ICCID) is not possible when this Operational Profile is in Enabled state

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | SW=0x6985 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #05 Error: Enable Profile (by ISD-P AID) not possible when an Operational Profile with a PPR1 is loaded

The purpose of this test is to ensure that it is NOT possible to enable an Operational Profile when there is another Operational Profile Enabled with the Policy Rule “Disabling of this Profile is not allowed”.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | No Profile is installed on the eUICC.  (this condition overrides the general initial condition defined in this test case) |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL4  NOTE: The PROFILE\_OPERATIONAL4 corresponds to <ISD\_P\_AID4>. | | |
| IC4 | Install PROFILE\_OPERATIONAL1  NOTE: The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. | | |
| IC5 | Enable PROFILE\_OPERATIONAL4 | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | SW=0x6985 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  #PROFILE\_INFO4\_ENABLED  }  SW=0x9000 |

Test Sequence #06 Error: Enable Profile (by ICCID) not possible with an Operational Profile with PPR1 is loaded

The purpose of this test is to ensure that it is NOT possible to enable an Operational Profile when there is another Operational Profile Enabled with the Policy Rule “Disabling of this Profile is not allowed”.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | No Profile is installed on the eUICC.  (this condition overrides the general initial condition defined in this test case) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step | Direction | | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | | |
| IC3 | Install PROFILE\_OPERATIONAL4 | | | |
| IC4 | Install PROFILE\_OPERATIONAL1 | | | |
| IC5 | Enable PROFILE\_OPERATIONAL4 | | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | | SW=0x6985 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA ( #GET\_PROFILES\_INFO\_ALL) | | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  #PROFILE\_INFO4\_ENABLED  }  SW=0x9000 |

Test Sequence #07 Error: Enable Profile by ISD-P AID without refreshFlag while proactive session is ongoing – catBusy supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC4 | Do not send FETCH command | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID2>,  FALSE)) | SW=0x9300 |
| 2 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1,  #PROFILE\_INFO2  }  SW=0x9000 |

Test Sequence #08 Error: Enable Profile by ICCID with refreshFLag set while proactive session is ongoing – catBusy supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC4 | Do not send FETCH command | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF2,  NO\_PARAM,  TRUE)) | SW=0x9300 |
| 2 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1,  #PROFILE\_INFO2  }  SW=0x9000 |

##### 4.2.21.2.3 TC\_eUICC\_ES10c.EnableProfile\_Case4

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |

Test Sequence #01 Nominal: Enable Profile by ISD-P AID and “refreshFlag” set when Device supports “UICC Reset”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | #R\_ENABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | S\_Device → eUICC | FETCH 'XX' | REFRESH Command (“UICC Reset”) |
| 3 | Repeat IC1 and IC2 | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 5 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 6 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

Test Sequence #02 Nominal: Enable Profile by ICCID and “refreshFlag” set when Device supports “UICC Reset”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | #R\_ENABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | S\_Device → eUICC | FETCH 'XX' | REFRESH Command (“UICC Reset”) |
| 3 | Repeat IC1 and IC2 | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 5 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 6 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

Test Sequence #03 Nominal: Enable Profile by ISD-P AID and “refreshFlag” set when Device supports “eUICC Profile State Change”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | #R\_ENABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | S\_Device → eUICC | FETCH 'XX' | REFRESH Command (“eUICC Profile State change”) |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | Execute IC1 from step 2 to step 4 | | |
| 5 | Repeat IC2 | | |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 7 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 8 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

Test Sequence #04 Nominal: Enable Profile by ICCID and “refreshFlag” set when Device supports “eUICC Profile State Change”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | #R\_ENABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | S\_Device → eUICC | FETCH 'XX' | REFRESH Command (“eUICC Profile State change”) |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | Execute IC1 from step 2 to step 4 | | |
| 5 | Repeat IC2 | | |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 7 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 8 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

Test Sequence #05 Nominal: Enable Profile by ISD-P AID and “refreshFlag” not set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  FALSE)) | #R\_ENABLE\_PROFILE\_OK  SW=0x9000 |
| 2 | S\_Device → eUICC | [TERMINAL\_PROFILE] | Toolkit initialization THEN SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 4 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 5 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

Test Sequence #06 Nominal: Enable Profile by ICCID and “refreshFlag” not set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  FALSE)) | #R\_ENABLE\_PROFILE\_OK  SW=0x9000 |
| 2 | S\_Device → eUICC | [TERMINAL\_PROFILE] | Toolkit initialization THEN SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 4 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 5 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

Test Sequence #07 Nominal: Enable Profile by ISD-P AID and “refreshFlag” set while proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC4 | Do not send FETCH command | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID2>,  TRUE)) | resp EnableProfileResponse ::= {  enableResult ok  }  SW=0x91YY |
| 2 | S\_Device 🡪eUICC | FETCH 'YY' | REFRESH Command (“UICC Reset”) |
| 3 | Repeat IC1 and IC2 | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED,  #PROFILE\_INFO2\_DISABLED  }  SW=0x9000 |
| 5 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 6 | S\_Device 🡪 eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF2  SW=0x9000 |

Test Sequence #08 Nominal: Enable Profile by ISD-P AID and “refreshFlag” not set while proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC4 | Do not send FETCH command | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID2>,  FALSE)) | resp EnableProfileResponse ::= {  enableResult ok  }  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED,  #PROFILE\_INFO2\_ENABLED  }  SW=0x9000 |
| 3 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 4 | S\_Device 🡪 eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF2  SW=0x9000 |

Test Sequence #09 Nominal: Enable Profile by ISD-P AID and “refreshFlag” set while proactive session is ongoing with Terminal Response outstanding – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| 1 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID2>,  TRUE)) | resp EnableProfileResponse ::= {  enableResult ok  }  SW=0x9000 |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x91YY |
| 4 | S\_Device 🡪eUICC | FETCH 'YY' | REFRESH Command (“UICC Reset”) |
| 5 | Repeat IC1 and IC2 | | |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED,  #PROFILE\_INFO2\_ENABLED  }  SW=0x9000 |
| 7 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 8 | S\_Device 🡪 eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF2  SW=0x9000 |

Test Sequence #10 Nominal: Enable Profile by ISD-P AID and “refreshFlag” not set while proactive session is ongoing with Terminal Response outstanding – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| 1 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID2>,  FALSE)) | resp EnableProfileResponse ::= {  enableResult ok  }  SW=0x9000 |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=any value except 91XX |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED,  #PROFILE\_INFO2\_ENABLED  }  SW=0x9000 |
| 5 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 6 | S\_Device 🡪 eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF2  SW=0x9000 |

##### 4.2.21.2.4 TC\_eUICC\_ES10c.EnableProfile\_ErrorCases\_Case4

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |

Test Sequence #01 Error: Enable Profile by an unknown ISD-P AID

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The Operational Profile identified by the ISD-P AID <ISD\_P\_AIDX> is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AIDX>,  TRUE)) | #R\_ENABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #02 Error: Enable Profile by an unknown ICCID

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The Operational Profile identified by the ICCID #ICCID\_OP\_PROFX is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROFX,  NO\_PARAM,  TRUE)) | #R\_ENABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #03 Error: Enable Profile (by ISD-P AID) is not possible when this Operational Profile is in Enable state

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | #R\_ENABLE\_PROFILE\_NOT\_DISABLE\_STATE  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #04 Error: Enable Profile (by ICCID) is not possible when this Operational Profile is in Enabled state

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | #R\_ENABLE\_PROFILE\_NOT\_DISABLE\_STATE  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #05 Error: Enable Profile (by ISD-P AID) not possible when an Operational Profile with PPR1 is loaded

The purpose of this test is to ensure that it is NOT possible to enable an Operational Profile when there is another Operational Profile Enabled with the Policy Rule “Disabling of this Profile is not allowed”.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | No Profile is installed on the eUICC.  (this condition overrides the general initial condition defined in this test case) |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL4  NOTE: The PROFILE\_OPERATIONAL4 corresponds to <ISD\_P\_AID4>. | | |
| IC4 | Install PROFILE\_OPERATIONAL1  NOTE: The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. | | |
| IC5 | Enable PROFILE\_OPERATIONAL4 | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA (  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | #R\_ENABLE\_PROFILE\_DISALLOWEDbyPOLICY  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  #PROFILE\_INFO4\_ENABLED  }  SW=0x9000 |

Test Sequence #06 Error: Enable Profile (by ICCID) not possible when an Operational Profile with PPR1 is loaded

The purpose of this test is to ensure that it is NOT possible to enable an Operational Profile when there is another Operational Profile Enabled with the Policy Rule “Disabling of this Profile is not allowed”.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | No Profile is installed on the eUICC.  (this condition overrides the general initial condition defined in this test case) |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL4 | | |
| IC4 | Install PROFILE\_OPERATIONAL1 | | |
| IC5 | Enable PROFILE\_OPERATIONAL4 | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | #R\_ENABLE\_PROFILE\_DISALLOWEDbyPOLICY  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  #PROFILE\_INFO4\_ENABLED  }  SW=0x9000 |

Test Sequence #07 Error: Enable Profile by ISD-P AID without refreshFlag while proactive session is ongoing – catBusy supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC4 | Do not send FETCH command | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID2>,  FALSE)) | resp EnableProfileResponse ::= {  enableResult catBusy  }  SW=0x9000 or 0x91XX |
| 2 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1,  #PROFILE\_INFO2  }  SW=0x9000 |

Test Sequence #08 Error: Enable Profile by ICCID with refreshFlag set while proactive session is ongoing – catBusy supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC4 | Do not send FETCH command | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF2,  NO\_PARAM,  TRUE)) | resp EnableProfileResponse ::= {  enableResult catBusy  }  SW=0x9000 or 0x91XX |
| 2 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1,  #PROFILE\_INFO2  }  SW=0x9000 |

##### 4.2.21.2.5 TC\_eUICC\_ES10c.EnableProfile\_ErrorCases\_Case4\_catBusySupported

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |

Test Sequence #01 Error: Enable Profile by ISD-P AID without refreshFlag while proactive session is ongoing

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC4 | Do not send FETCH command | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID2>,  FALSE)) | resp EnableProfileResponse ::= {  enableResult catBusy  }  SW= 0x91XX |
| 2 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1,  #PROFILE\_INFO2  }  SW=0x9000 |

Test Sequence #02 Error: Enable Profile by ICCID with refreshFlag set while proactive session is ongoing

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC4 | Do not send FETCH command | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF2,  NO\_PARAM,  TRUE)) | resp EnableProfileResponse ::= {  enableResult catBusy  }  SW= 0x91XX |
| 2 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1,  #PROFILE\_INFO2  }  SW=0x9000 |

##### 4.2.21.2.6 TC\_eUICC\_ES10c.EnableProfile\_Case4\_catBusyNotSupported

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |

Test Sequence #01: Enable Profile by ISD-P AID without refreshFlag while proactive session is ongoing

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC4 | Do not send FETCH command | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID2>,  FALSE)) | resp EnableProfileResponse ::= {  #R\_ENABLE\_PROFILE\_OK  }  SW=0x9000 |
| 2 | S\_Device → eUICC | [TERMINAL\_PROFILE] | Toolkit initialization THEN SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED, #PROFILE\_INFO2\_ENABLED  }  SW=0x9000 |
| 4 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 5 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF2  SW=0x9000 |

Test Sequence #02: Enable Profile by ICCID with refreshFlag set while proactive session is ongoing

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC4 | Do not send FETCH command | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF2,  NO\_PARAM,  TRUE)) | #R\_ENABLE\_PROFILE\_OK  SW= 0x91YY  Note: 91YY MAY be different from 91XX in IC3 |
| 2 | S\_Device → eUICC | FETCH 'YY' | REFRESH Command (“UICC Reset”) |
| 3 | Repeat IC1 and IC2 | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED,  #PROFILE\_INFO2\_ENABLED  }  SW=0x9000 |
| 5 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 6 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF2  SW=0x9000 |

##### 4.2.21.2.7 TC\_eUICC\_ES10c.EnableProfile\_ErrorCases\_Case3\_catBusySupported

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |

Test Sequence #01 Error: Enable Profile by ISD-P AID without refreshFlag while proactive session is ongoing

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC4 | Do not send FETCH command | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID2>,  FALSE)) | SW=0x9300 |
| 2 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1,  #PROFILE\_INFO2  }  SW=0x9000 |

Test Sequence #02 Error: Enable Profile by ICCID with refreshFLag set while proactive session is ongoing

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC4 | Do not send FETCH command | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF2,  NO\_PARAM,  TRUE)) | SW=0x9300 |
| 2 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1,  #PROFILE\_INFO2  }  SW=0x9000 |

##### 4.2.21.2.8 TC\_eUICC\_ES10c.EnableProfile\_Case4\_MEPA1

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | eUICC in MEP mode |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |

Test Sequence #01 Nominal: Enable 1st Profile by ISD-P AID and “refreshFlag” set when Device supports “UICC Reset”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify <MEP\_MODE> = ’01’,  Verify <MEP\_LSI\_OPTION> =  #IUT\_MEP\_LSI\_OPTIONS,  Verify <MEP\_MAX\_LSIS> <=  #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE, 1)) | #R\_ENABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  1,  “UICC Reset”  ) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_FIRST\_PROFILE | | |
| 4 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 6 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 7 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 8 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

Test Sequence #02 Nominal: Enable Profile by ICCID and “refreshFlag” set when Device supports “UICC Reset”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = ’01’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE,  1)) | #R\_ENABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  1,  “UICC Reset”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_FIRST\_PROFILE | | |
| 4 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 6 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 7 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 8 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

Test Sequence #03 Nominal: Enable Profile by ISD-P AID and “refreshFlag” set when Device supports “eUICC Profile State Change”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify <MEP\_MODE> = ’01’,  Verify <MEP\_LSI\_OPTION> =  #IUT\_MEP\_LSI\_OPTIONS,  Verify <MEP\_MAX\_LSIS> <=  #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE,  1)) | #R\_ENABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  1,  “eUICC Profile State Change”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP\_EN\_DS\_FIRST\_PROFILE | | |
| 4 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 6 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 7 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 8 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

Test Sequence #04 Nominal: Enable Profile by ICCID and “refreshFlag” set when Device supports “eUICC Profile State Change”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify  ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = ’01’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE,  1)) | #R\_ENABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  1,  “eUICC Profile State Change”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP\_EN\_DS\_FIRST\_PROFILE | | |
| 4 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 6 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 7 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 8 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

Test Sequence #05 Nominal: Enable Profile by ISD-P AID and “refreshFlag” not set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify  ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> =’01’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  NO\_PARAM,  <ISD\_P\_AID1>,  FALSE,  1)) | #R\_ENABLE\_PROFILE\_OK  SW=0x9000 |
| 2 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 3 | S\_Device → eUICC | [TERMINAL\_PROFILE\_LSI\_COMMAND] | Toolkit initialization THEN SW=0x9000 |
| 4 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 6 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 7 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 8 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

Test Sequence #06 Nominal: Enable Profile by ICCID and “refreshFlag” not set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify  ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = ’01’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  #ICCID\_OP\_PROF1,  NO\_PARAM,  FALSE,  1)) | #R\_ENABLE\_PROFILE\_OK  SW=0x9000 |
| 2 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 3 | S\_Device → eUICC | [TERMINAL\_PROFILE\_LSI\_COMMAND] | Toolkit initialization THEN SW=0x9000 |
| 4 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 6 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 7 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 8 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

Test Sequence #07 Nominal: Enable Profile by ISD-P AID and “refreshFlag” set while proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1 |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify  ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = ’01’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC8 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  NO\_PARAM,  <ISD\_P\_AID2>,  TRUE,  1)) | resp EnableProfileResponse ::= {  enableResult ok  }  SW=0x91YY |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  1,  “UICC Reset”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_FIRST\_PROFILE | | |
| 4 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO (  <NO\_PARAM>,  <NO\_PARAM>) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED, #PROFILE\_INFO2\_ENABLED  }  SW=0x9000 |
| 6 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 7 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 8 | S\_Device 🡪 eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF2  SW=0x9000 |

Test Sequence #08 Nominal: Enable Profile by ISD-P AID and “refreshFlag” not set while proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify  ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP(  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> =’01’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC8 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  NO\_PARAM,  <ISD\_P\_AID2>,  FALSE,  1)) | resp EnableProfileResponse ::= {  enableResult ok  }  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED,  #PROFILE\_INFO2\_ENABLED  }  SW=0x9000 |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 4 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 5 | S\_Device 🡪 eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF2  SW=0x9000 |

Test Sequence #09 Nominal: Enable 2nd Profile by ISD-P AID and “refreshFlag” set when Device supports “UICC Reset”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  NO\_PARAM,  <ISD\_P\_AID2>,  TRUE,  2)) | #R\_ENABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  <MEP\_MODE>,  2,  “UICC Reset”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 4 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |
| 6 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| 7 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 8 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF2  SW=0x9000 |

Test Sequence #10 Nominal: Enable 2nd Profile by ICCID and “refreshFlag” set when Device supports “UICC Reset”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1 . |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to #ICCID\_OP\_PROF1 |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to #ICCID\_OP\_PROF2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  #ICCID\_OP\_PROF2,  NO\_PARAM,  TRUE,  2)) | #R\_ENABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  <MEP\_MODE>,  2,  “UICC Reset”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 4 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |
| 6 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| 7 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 8 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF2  SW=0x9000 |

Test Sequence #11 Nominal: Enable 2nd Profile by ISD-P AID and “refreshFlag” set when Device supports “eUICC Profile State Change”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  NO\_PARAM,  <ISD\_P\_AID2>,  TRUE,  2)) | #R\_ENABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  <MEP\_MODE>,  0,  “eUICC Profile State Change”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP \_EN\_DS\_SECOND\_PROFILE | | |
| 4 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |
| 6 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| 7 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 8 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF2  SW=0x9000 |

Test Sequence #12 Nominal: Enable 2nd Profile by ICCID and “refreshFlag” set when Device supports “eUICC Profile State Change”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to #ICCID\_OP\_PROF1 |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to #ICCID\_OP\_PROF2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  #ICCID\_OP\_PROF2,  NO\_PARAM,  TRUE,  2)) | #R\_ENABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  <MEP\_MODE>,  2,  “eUICCProfileStateChange”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 4 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |
| 6 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| 7 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 8 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF2  SW=0x9000 |

Test Sequence #13 Nominal: Enable 2nd Profile by ISD-P AID and “refreshFlag” not set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  NO\_PARAM,  <ISD\_P\_AID2>,  FALSE,  2)) | #R\_ENABLE\_PROFILE\_OK  SW=0x9000 |
| 2 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO (  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |
| 5 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| 6 | S\_Device → eUICC | [TERMINAL\_PROFILE\_LSI\_COMMAND] | Toolkit initialization THEN SW=0x9000 |
| 7 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 8 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF2  SW=0x9000 |

Test Sequence #14 Nominal: Enable 2nd Profile by ICCID and “refreshFlag” not set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to #ICCID\_OP\_PROF1 |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to #ICCID\_OP\_PROF2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  #ICCID\_OP\_PROF2,  NO\_PARAM,  FALSE,  2)) | #R\_ENABLE\_PROFILE\_OK  SW=0x9000 |
| 2 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |
| 5 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| 6 | S\_Device → eUICC | [TERMINAL\_PROFILE\_LSI\_COMMAND] | Toolkit initialization THEN SW=0x9000 |
| 7 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 8 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF2  SW=0x9000 |

Test Sequence #15 Nominal: Enable 3rd Profile by ISD-P AID and “refreshFlag” set while proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 2. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 corresponds to <ISD\_P\_AID3>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC8 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  NO\_PARAM,  <ISD\_P\_AID3>,  TRUE,  2)) | resp EnableProfileResponse ::= {  enableResult ok  }  SW=0x91YY |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  2,  “UICC Reset”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 4 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO3\_ENABLED;  }  SW=0x9000 |
| 6 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| 7 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 8 | S\_Device 🡪 eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF3  SW=0x9000 |

Test Sequence #16 Nominal: Enable 3rd Profile by ISD-P AID and “refreshFlag” set while proactive session is ongoing with Terminal Response outstanding – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 2. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 corresponds to <ISD\_P\_AID3>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | S\_Device → eUICC | FETCH ‘XX’ | SMS POR received  SCP80 response status code equal to 0x00 – POR OK and waiting for Terminal Response |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC8 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  NO\_PARAM,  <ISD\_P\_AID3>,  TRUE,  1)) | resp EnableProfileResponse ::= {  enableResult ok  }  SW=0x91YY |
| 2 | S\_Device 🡪eUICC | FETCH 'YY' | LSI COMMAND (“Proactive Session Request”)  Verify <LSI\_COMMAND\_ACTION> = “Proactive session request” and <LSI\_NUMBER> = 1 |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING( <LSI\_NUMBER>) | | |
| 4 | S\_Device 🡪eUICC | STATUS command | SW=0x9000 |
| 5 | S\_Device 🡪eUICC | TERMINAL RESPONSE | SW=0x91ZZ |
| 6 | S\_Device 🡪eUICC | FETCH 'ZZ' | REFRESH Command (“UICC Reset”) |
| 7 | S\_Device → eUICC | [SELECT\_MF] | FCP Template present  SW=0x9000 |
| 8 | S\_Device → eUICC | [TERMINAL\_CAPABILITY\_LPAd] | SW=0x9000 |
| 9 | S\_Device → eUICC | [TERMINAL\_PROFILE\_LSI\_COMMAND] | Toolkit initialization THEN SW=0x9000 |
| 10 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 11 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO3\_ENABLED;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |
| 12 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 13 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 14 | S\_Device 🡪 eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF3  SW=0x9000 |

Test Sequence #17 Nominal: Enable 3rd Profile by ISD-P AID and “refreshFlag” set while 2 proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 2. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 corresponds to <ISD\_P\_AID3>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC8 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91YY |
| IC9 | Do not send FETCH command | | |
| IC10 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC11 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → Euicc | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  NO\_PARAM,  <ISD\_P\_AID3>,  TRUE,  1)) | resp EnableProfileResponse ::= {  enableResult ok  }  SW=0x91ZZ |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  1,  “UICC Reset”) | | |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| 4 | S\_Device 🡪eUICC | FETCH ‘XX’ | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 5 | S\_Device 🡪eUICC | TERMINAL RESPONSE |  |
| 6 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO3\_ENABLED;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |
| 8 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 9 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 10 | S\_Device 🡪 eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF3  SW=0x9000 |

Test Sequence #18 Nominal: Enable 3rd Profile by ISD-P AID and “refreshFlag” set while 2 proactive session is ongoing with Terminal Response outstanding – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 2. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 corresponds to <ISD\_P\_AID3>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | S\_Device → eUICC | FETCH ‘XX’ | SMS POR received  SCP80 response status code equal to 0x00 – POR OK and waiting for Terminal Response |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC8 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91YY |
| IC9 | S\_Device → eUICC | FETCH ‘YY’ | SMS POR received  SCP80 response status code equal to 0x00 – POR OK and waiting for Terminal Response |
| IC10 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC11 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  NO\_PARAM,  <ISD\_P\_AID3>,  TRUE,  1)) | resp EnableProfileResponse ::= {  enableResult ok  }  SW=0x91ZZ |
| 2 | S\_Device 🡪eUICC | FETCH 'ZZ' | LSI COMMAND (“ProactiveSessionREquest”)  Verify <LSI\_COMMAND\_ACTION> = “Proactive session request” and <LSI\_NUMBER> = 1 |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(<LSI\_NUMBER>) | | |
| 4 | S\_Device 🡪eUICC | STATUS command | SW=0x9000 |
| 5 | S\_Device 🡪eUICC | TERMINAL RESPONSE | SW=0x91KK |
| 6 | S\_Device 🡪eUICC | FETCH 'KK' | REFRESH Command (“UICC Reset”) |
| 7 | S\_Device → eUICC | [SELECT\_MF] | FCP Template present  SW=0x9000 |
| 8 | S\_Device → eUICC | [TERMINAL\_CAPABILITY\_LPAd] | SW=0x9000 |
| 9 | S\_Device → eUICC | [TERMINAL\_PROFILE\_LSI\_COMMAND] | Toolkit initialization THEN SW=0x9000 |
| 10 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| 11 | S\_Device 🡪eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 12 | S\_Device | PROC\_MEP\_LSI\_MULTIPLEXING(0) |  |
| 13 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO3\_ENABLED;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |
| 14 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 15 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 16 | S\_Device 🡪 eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF3  SW=0x9000 |

##### 4.2.21.2.9 TC\_eUICC\_ES10c.EnableProfile\_ErrorCases\_Case4\_MEPA1

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | eUICC in MEP mode |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |

Test Sequence #01 Error: Enable Profile by an unknown ISD-P AID

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The Operational Profile identified by the ISD-P AID <ISD\_P\_AIDX> is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify  ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> =’01’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  NO\_PARAM,  <ISD\_P\_AIDX>,  TRUE,  1)) | #R\_ENABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #02 Error: Enable Profile by an unknown ICCID

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The Operational Profile identified by the ICCID #ICCID\_OP\_PROFX is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify  ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = ’01’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  #ICCID\_OP\_PROFX,  NO\_PARAM,  TRUE,  1)) | #R\_ENABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #03 Error: Enable Profile (by ISD-P AID) is not possible when this Operational Profile is in Enable state

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify  ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> =’01’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE,  1)) | #R\_ENABLE\_PROFILE\_NOT\_DISABLE\_STATE  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #04 Error: Enable Profile (by ICCID) is not possible when this Operational Profile is in Enabled state

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify  ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> =’01’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE,  1)) | #R\_ENABLE\_PROFILE\_NOT\_DISABLE\_STATE  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #05 Error: Enable Profile by ISD-P AID without refreshFlag while proactive session is ongoing – catBusy supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify  ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> =’01’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC8 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  NO\_PARAM,  <ISD\_P\_AID2>,  FALSE,  1)) | resp EnableProfileResponse ::= {  enableResult catBusy  }  SW=0x9000 or 0x91XX |
| 2 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 3 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 4 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 5 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1,  #PROFILE\_INFO2  }  SW=0x9000 |

Test Sequence #06 Error: Enable Profile by ICCID with refreshFlag set while proactive session is ongoing – catBusy supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify  ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = ’01’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC8 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  #ICCID\_OP\_PROF2,  NO\_PARAM,  TRUE,  1)) | resp EnableProfileResponse ::= {  enableResult catBusy  }  SW=0x9000 or 0x91XX |
| 2 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 3 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 4 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 5 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO (  <NO\_PARAM>,  <NO\_PARAM>) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1,  #PROFILE\_INFO2  }  SW=0x9000 |

Test Sequence #07 Error: Enable 2nd Profile by an unknown ISD-P AID where one profile is already enabled

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The Operational Profile identified by the ISD-P AID <ISD\_P\_AIDX> is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  NO\_PARAM,  <ISD\_P\_AIDX>,  TRUE,  2)) | #R\_ENABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  }  SW=0x9000 |

Test Sequence #08 Error: Enable 2nd Profile by an unknown ICCID where one profile is already enabled

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to #ICCID\_OP\_PROF1. |
| eUICC | The Operational Profile identified by the ICCID #ICCID\_OP\_PROFX is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  #ICCID\_OP\_PROFX,  NO\_PARAM,  TRUE,  2)) | #R\_ENABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  }  SW=0x9000 |

Test Sequence #09 Error: Enable 2nd Profile (by ISD-P AID) is not possible when this Operational Profile is in Enable state at other LSI

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 2. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE,  2)) | #R\_ENABLE\_PROFILE\_NOT\_DISABLE\_STATE  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  NO\_PARAM,  <ISD\_P\_AID2>,  TRUE,  1)) | #R\_ENABLE\_PROFILE\_NOT\_DISABLE\_STATE  SW=0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |

Test Sequence #10 Error: Enable 2nd Profile (by ICCID) is not possible when this Operational Profile is in Enabled state at other LSI

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to #ICCID\_OP\_PROF1 |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 2. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to #ICCID\_OP\_PROF2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE,  2)) | #R\_ENABLE\_PROFILE\_NOT\_DISABLE\_STATE  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  #ICCID\_OP\_PROF2,  NO\_PARAM,  TRUE,  1)) | #R\_ENABLE\_PROFILE\_NOT\_DISABLE\_STATE  SW=0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF2,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |

Test Sequence #11 Error: Enable 2nd Profile by ISD-P AID and “refreshFlag” set while 2 proactive session is ongoing – catBusy supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 2. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 corresponds to <ISD\_P\_AID3>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC8 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91YY |
| IC9 | Do not send FETCH command | | |
| IC10 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC11 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  NO\_PARAM,  <ISD\_P\_AID3>,  TRUE,  1)) | resp EnableProfileResponse ::= {  enableResult catBusy  }  SW= 9000 |
| 2 | Repeat IC7 | | |
| 3 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 4 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 5 | Repeat IC4 | | |
| 6 | S\_Device → eUICC | FETCH 'YY' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 7 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 8 | Repeat IC10 | | |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |

Test Sequence #12 Error: Enable 2nd Profile (by ISD-P AID) is not possible when CAT is not initialized for targetPort

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_ONE\_LSI\_FOR\_ENABLED\_PROFILE | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE\_MEP\_A1(  NO\_PARAM,  <ISD\_P\_AID3>,  TRUE,  2)) | resp EnableProfileResponse ::= {  enableResult commandError  }  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1,  #PROFILE\_INFO2  }  SW=0x9000 |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 4 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 5 | S\_Device 🡪 eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

##### 4.2.21.2.10 TC\_eUICC\_ES10c.EnableProfile\_Case4\_MEPA2

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | eUICC in MEP mode |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |

Test Sequence #01 Nominal: Enable Profile by ISD-P AID and “refreshFlag” not set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify  ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> =’02’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  FALSE)) | #R\_ENABLE\_PROFILE\_MEP\_A2\_OK  SW=0x9000  Verify < MEP-A2\_TARGET\_ESIM\_PORT> = 1 OR 2 |
| 2 | PROC\_MEP\_LSI\_MULTIPLEXING(<MEP-A2\_TARGET\_ESIM\_PORT>) | | |
| 3 | S\_Device → eUICC | [TERMINAL\_PROFILE\_LSI\_COMMAND] | Toolkit initialization THEN SW=0x9000 |
| 4 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 5 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |
| 6 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM  )) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #02 Nominal: Enable Profile by ICCID and “refreshFlag” not set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify  ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = ‘02’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  FALSE)) | #R\_ENABLE\_PROFILE\_MEP\_A2\_OK  SW=0x9000  Verify < MEP-A2\_TARGET\_ESIM\_PORT> = 1 OR 2 |
| 2 | PROC\_MEP\_LSI\_MULTIPLEXING(<MEP-A2\_TARGET\_ESIM\_PORT>) | | |
| 3 | S\_Device → eUICC | [TERMINAL\_PROFILE\_LSI\_COMMAND] | Toolkit initialization THEN SW=0x9000 |
| 4 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 6 | PROC\_MEP\_LSI\_MULTIPLEXING(<MEP-A2\_TARGET\_ESIM\_PORT>) | | |
| 7 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 8 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

Test Sequence #03 Nominal: Enable 2nd Profile by ISD-P AID and “refreshFlag” not set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID2>,  FALSE)) | #R\_ENABLE\_PROFILE\_MEP\_A2\_OK  SW=0x9000  Verify <MEP-A2\_TARGET\_ESIM\_PORT> = 1 OR 2 |
| 2 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |
| 5 | PROC\_MEP\_LSI\_MULTIPLEXING(<MEP-A2\_TARGET\_ESIM\_PORT>) | | |
| 6 | S\_Device → eUICC | [TERMINAL\_PROFILE\_LSI\_COMMAND] | Toolkit initialization THEN SW=0x9000 |
| 7 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 8 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF2  SW=0x9000 |

Test Sequence #04 Nominal: Enable 2nd Profile by ICCID and “refreshFlag” not set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to #ICCID\_OP\_PROF1 |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to #ICCID\_OP\_PROF2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF2,  NO\_PARAM,  FALSE)) | #R\_ENABLE\_PROFILE\_MEP\_A2\_OK  SW=0x9000  Verify <MEP-A2\_TARGET\_ESIM\_PORT> =1 OR 2 |
| 2 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO (  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |
| 5 | PROC\_MEP\_LSI\_MULTIPLEXING(<MEP-A2\_TARGET\_ESIM\_PORT>) | | |
| 6 | S\_Device → eUICC | [TERMINAL\_PROFILE\_LSI\_COMMAND] | Toolkit initialization THEN SW=0x9000 |
| 7 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 8 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF2  SW=0x9000 |

##### 4.2.21.2.11 TC\_eUICC\_ES10c.EnableProfile\_ErrorCases\_Case4\_MEPA2

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | eUICC in MEP mode |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |

Test Sequence #01 Error: Enable Profile by an unknown ISD-P AID

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The Operational Profile identified by the ISD-P AID <ISD\_P\_AIDX> is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify  ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = ‘02’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE (  NO\_PARAM,  <ISD\_P\_AIDX>,  FALSE)) | #R\_ENABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #02 Error: Enable Profile by an unknown ICCID

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The Operational Profile identified by the ICCID #ICCID\_OP\_PROFX is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify  ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = ‘02’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROFX,  NO\_PARAM,  FALSE)) | #R\_ENABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #03 Error: Enable Profile (by ISD-P AID) is not possible when this Operational Profile is in Enabled state

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC as MEPA2 mode. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify  ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = ‘02’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  FALSE)) | #R\_ENABLE\_PROFILE\_NOT\_DISABLE\_STATE  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #04 Error: Enable Profile (by ICCID) is not possible when this Operational Profile is in Enabled state

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC as MEPA2 mode. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify  ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = ‘02’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE (  #ICCID\_OP\_PROF1,  NO\_PARAM,  FALSE)) | #R\_ENABLE\_PROFILE\_NOT\_DISABLE\_STATE  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #05 Error: Enable Profile by ISD-P AID without refreshFlag while proactive session is ongoing – catBusy supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC as MEPA2 mode. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify  ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = ‘02',  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(<MEP-A2\_TARGET\_ESIM\_PORT>) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC8 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID2>,  FALSE)) | resp EnableProfileResponse ::= {  enableResult catBusy  }  SW=0x9000 or 0x91XX |
| 2 | PROC\_MEP\_LSI\_MULTIPLEXING(<MEP-A2\_TARGET\_ESIM\_PORT>) | | |
| 3 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 4 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 5 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO (  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1,  #PROFILE\_INFO2  }  SW=0x9000 |

Test Sequence #06 Error: Enable 2ndProfile by an unknown ISD-P AID where one profile is already enabled

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC as MEPA2 mode. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The Operational Profile identified by the ISD-P AID <ISD\_P\_AIDX> is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE (  NO\_PARAM,  <ISD\_P\_AIDX>,  FALSE)) | #R\_ENABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  }  SW=0x9000 |

Test Sequence #07 Error: Enable 2ndProfile by an unknown ICCID where one profile is already enabled

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC as MEPA2 mode. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to #ICCID\_OP\_PROF1. |
| eUICC | The Operational Profile identified by the ICCID #ICCID\_OP\_PROFX is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE (  #ICCID\_OP\_PROFX,  NO\_PARAM,  FALSE)) | #R\_ENABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  }  SW=0x9000 |

Test Sequence #08 Error: Enable 2ndProfile (by ISD-P AID) is not possible when this Operational Profile is in Enable state at other LSI

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port X. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port Y. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE (  NO\_PARAM,  <ISD\_P\_AID1>,  FALSE)) | #R\_ENABLE\_PROFILE\_NOT\_DISABLE\_STATE  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE (  NO\_PARAM,  <ISD\_P\_AID2>,  FALSE,  1)) | #R\_ENABLE\_PROFILE\_NOT\_DISABLE\_STATE  SW=0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |

Test Sequence #09 Error: Enable 2ndProfile (by ICCID) is not possible when this Operational Profile is in Enabled state at other LSI

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port X. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to #ICCID\_OP\_PROF1 |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port Y. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to #ICCID\_OP\_PROF2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  FALSE,  2)) | #R\_ENABLE\_PROFILE\_NOT\_DISABLE\_STATE  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE (  #ICCID\_OP\_PROF2,  NO\_PARAM,  FALSE,  1)) | #R\_ENABLE\_PROFILE\_NOT\_DISABLE\_STATE  SW=0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF2,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |

Test Sequence #10 Error: Enable 2ndProfile by ISD-P AID and “refreshFlag” not set while 2 proactive session is ongoing – catBusy supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port X. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port Y. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 corresponds to <ISD\_P\_AID3>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(Y) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(X) | | |
| IC8 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91YY |
| IC9 | Do not send FETCH command | | |
| IC10 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC11 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID3>,  FALSE,  1)) | resp EnableProfileResponse ::= {  enableResult catBusy  }  SW= 9000 |
| 2 | Repeat IC7 | | |
| 3 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 4 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 5 | Repeat IC4 | | |
| 6 | S\_Device → eUICC | FETCH 'YY' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 7 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 8 | Repeat IC10 | | |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO (  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |

##### 4.2.21.2.12 TC\_eUICC\_ES10c.EnableProfile\_Case4\_MEPB

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | eUICC in MEP mode |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |

Test Sequence #01 Nominal: Enable 1st Profile by ISD-P AID and “refreshFlag” set when Device supports “UICC Reset”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify <MEP\_MODE> = ‘03’,  <MEP\_LSI\_OPTION> =  #IUT\_MEP\_LSI\_OPTIONS,  <MEP\_MAX\_LSIS> <=  #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE (  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)  ) | #R\_ENABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  NO\_PARAM,  “UICC Reset”  ) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_FIRST\_PROFILE | | |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 6 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 7 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

Test Sequence #02 Nominal: Enable Profile by ICCID and “refreshFlag” set when Device supports “UICC Reset”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = ‘03’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE (  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | #R\_ENABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  NO\_PARAM,  “UICC Reset”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_FIRST\_PROFILE | | |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 6 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 7 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

Test Sequence #03 Nominal: Enable Profile by ISD-P AID and “refreshFlag” set when Device supports “eUICC Profile State Change”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify <MEP\_MODE> = ‘03’,  Verify <MEP\_LSI\_OPTION> =  #IUT\_MEP\_LSI\_OPTIONS,  Verify <MEP\_MAX\_LSIS> <=  #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | #R\_ENABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  <MEP\_MODE>,  0,  “eUICC Profile State change”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP\_EN\_DS\_FIRST\_PROFILE | | |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 6 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 7 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

Test Sequence #04 Nominal: Enable Profile by ICCID and “refreshFlag” set when Device supports “eUICC Profile State Change”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify  ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = ‘03’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE (  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | #R\_ENABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  <MEP\_MODE>,  0,  “eUICC Profile State Change”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP\_EN\_DS\_FIRST\_PROFILE | | |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 6 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 7 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

Test Sequence #05 Nominal: Enable Profile by ISD-P AID and “refreshFlag” not set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify  ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = ‘03’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE (  NO\_PARAM,  <ISD\_P\_AID1>,  FALSE)) | #R\_ENABLE\_PROFILE\_OK  SW=0x9000 |
| 2 | S\_Device → eUICC | [TERMINAL\_PROFILE\_LSI\_COMMAND] | Toolkit initialization THEN SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 4 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 5 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

Test Sequence #06 Nominal: Enable Profile by ICCID and “refreshFlag” not set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify  ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = ‘03’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE (  #ICCID\_OP\_PROF1,  NO\_PARAM,  FALSE)) | #R\_ENABLE\_PROFILE\_OK  SW=0x9000 |
| 2 | S\_Device → eUICC | [TERMINAL\_PROFILE\_LSI\_COMMAND] | Toolkit initialization THEN SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 4 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 5 | S\_Device → Euicc | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

Test Sequence #07 Nominal: Enable Profile by ISD-P AID and “refreshFlag” set whileproactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify  ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = ‘03’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC5 | Do not send FETCH command | | |
| IC6 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE (  NO\_PARAM,  <ISD\_P\_AID2>,  TRUE)) | resp EnableProfileResponse ::= {  enableResult ok  }  SW=0x91YY |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  <MEP\_MODE>,  0,  “UICC Reset”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_FIRST\_PROFILE | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED, #PROFILE\_INFO2\_ENABLED  } |
| 5 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 6 | S\_Device 🡪 Euicc | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF2  SW=0x9000 |

Test Sequence #08 Nominal: Enable Profile by ISD-P AID and “refreshFlag” not set while proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify  ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = ‘03’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC5 | Do not send FETCH command | | |
| IC6 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE (  NO\_PARAM,  <ISD\_P\_AID2>,  FALSE)) | resp EnableProfileResponse ::= {  enableResult ok  }  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED,  #PROFILE\_INFO2\_ENABLED  }  SW=0x9000 |
| 3 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 4 | S\_Device 🡪 eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF2  SW=0x9000 |

Test Sequence #09 Nominal: Enable 2nd Profile by ISD-P AID and “refreshFlag” set when Device supports “UICC Reset”

The purpose of this sequence is to make sure that the MEP-capable eUICC can have two Profiles Enabled on two different LSIs.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC5 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC6 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE (  NO\_PARAM,  <ISD\_P\_AID2>,  TRUE)) | #R\_ENABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  <MEP\_MODE>,  1,  “UICC Reset”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_MEPB;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |
| 6 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 7 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF2  SW=0x9000 |

Test Sequence #10 Nominal: Enable 2nd Profile by ICCID and “refreshFlag” set when Device supports “UICC Reset”

The purpose of this test sequence is to make sure that the MEP-capable eUICC can have two Profiles Enabled on two different LSIs.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to #ICCID\_OP\_PROF1 |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to #ICCID\_OP\_PROF2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE (  #ICCID\_OP\_PROF2,  NO\_PARAM,  TRUE)) | #R\_ENABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  1,  “UICC Reset”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 4 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 5 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_ENABLED\_MEPB;  }  SW=0x9000 |
| 7 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 8 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 9 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF2  SW=0x9000 |

Test Sequence #11 Nominal: Enable 2nd Profile by ISD-P AID and “refreshFlag” set when Device supports “eUICC Profile State Change”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE (  NO\_PARAM,  <ISD\_P\_AID2>,  TRUE)) | #R\_ENABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  1,  “eUICC Profile State Change”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP \_EN\_DS\_SECOND\_PROFILE | | |
| 4 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 5 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_ENABLED\_MEPB;  }  SW=0x9000 |
| 7 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| 8 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 9 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF2  SW=0x9000 |

Test Sequence #12 Nominal: Enable 2nd Profile by ICCID and “refreshFlag” set when Device supports “eUICC Profile State Change”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to #ICCID\_OP\_PROF1 |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to #ICCID\_OP\_PROF2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE (  #ICCID\_OP\_PROF2,  NO\_PARAM,  TRUE,  <MEP\_MODE>,  1,  NO\_PARAM)) | #R\_ENABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | S\_Device → eUICC | PROC\_MEP\_REFRESH\_EN\_DS(  1,  “eUICCProfileStateChange”) | REFRESH Command (“eUICC Profile State Change”) |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE eUICCProfileStateChanged\_MEP \_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 4 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 5 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_MEP;  #PROFILE\_INFO2\_MEPB;  }  SW=0x9000 |
| 7 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 8 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 9 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF2  SW=0x9000 |

Test Sequence #13 Nominal: Enable 2nd Profile by ISD-P AID and “refreshFlag” not set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID2>,  FALSE)) | #R\_ENABLE\_PROFILE\_OK  SW=0x9000 |
| 2 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_MEPB;  }  SW=0x9000 |
| 6 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 7 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 8 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF2  SW=0x9000 |

Test Sequence #14 Nominal: Enable 2nd Profile by ICCID and “refreshFlag” not set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to #ICCID\_OP\_PROF1 |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to #ICCID\_OP\_PROF2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF2,  NO\_PARAM,  FALSE)) | #R\_ENABLE\_PROFILE\_OK  SW=0x9000 |
| 2 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_ENABLED\_MEPB;  }  SW=0x9000 |
| 6 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 7 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 8 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF2  SW=0x9000 |

Test Sequence #15 Nominal: Enable 3rd Profile by ISD-P AID and “refreshFlag” set while proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 corresponds to <ISD\_P\_AID3>. |
| eUICC | The Nickname of the PROFILE\_OPERATIONAL3 is equal to #NICKNAME3. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID3>,  TRUE)) | resp EnableProfileResponse ::= {  enableResult ok  }  SW=0x91YY |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  1,  UICC Reset) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 4 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 5 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2;  #PROFILE\_INFO3\_MEPB;  }  SW=0x9000 |
| 7 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 8 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 9 | S\_Device 🡪 eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF3  SW=0x9000 |

Test Sequence #16 Nominal: Enable 3rd Profile by ISD-P AID and “refreshFlag” set while 2 proactive session is ongoing – catBusy not supported

The purpose of this test sequence is to verify that profile switching of one target port (here Port 1) does not impact a proactive command on another eSIM Port (here Port 0).

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 corresponds to <ISD\_P\_AID3>. |
| eUICC | The Nickname of the PROFILE\_OPERATIONAL3 is equal to #NICKNAME3. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC8 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91YY |
| IC9 | Do not send FETCH command | | |
| IC10 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC11 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID3>,  TRUE)) | resp EnableProfileResponse ::= {  enableResult ok  }  SW=0x91ZZ |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  1,  “UICC Reset”) | | |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 4 | S\_Device 🡪eUICC | FETCH 'YY' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 5 | S\_Device 🡪eUICC | TERMINAL RESPONSE |  |
| 6 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 7 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2;  #PROFILE\_INFO3\_MEPB  }  SW=0x9000 |
| 9 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 10 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 11 | S\_Device 🡪 eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF3  SW=0x9000 |

##### 4.2.21.2.13 TC\_eUICC\_ES10c.EnableProfile\_ErrorCases\_Case4\_MEPB

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | eUICC in MEP mode |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |

Test Sequence #01 Error: Enable Profile by an unknown ISD-P AID

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The Operational Profile identified by the ISD-P AID <ISD\_P\_AIDX> is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify  ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = ‘03’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AIDX>,  TRUE)) | #R\_ENABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #02 Error: Enable Profile by an unknown ICCID

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The Operational Profile identified by the ICCID #ICCID\_OP\_PROFX is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify  ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = ‘03’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROFX,  NO\_PARAM,  TRUE)) | #R\_ENABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #03 Error: Enable Profile (by ISD-P AID) is not possible when this Operational Profile is in Enabled state

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify  ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = ‘03’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | #R\_ENABLE\_PROFILE\_NOT\_DISABLE\_STATE  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #04 Error: Enable Profile (by ICCID) is not possible when this Operational Profile is in Enabled state

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify  ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = ‘03’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | #R\_ENABLE\_PROFILE\_NOT\_DISABLE\_STATE  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #05 Error: Enable Profile by ISD-P AID with refreshFlag not set while proactive session is ongoing – catBusy supported

The purpose of this test sequence is to verify that if a proactive session is still ongoing on the target Port and catBusy is supported, the Enable command is rejected.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify  ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = ‘03’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC5 | Do not send FETCH command | | |
| IC6 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE (  NO\_PARAM,  <ISD\_P\_AID2>,  FALSE)) | resp EnableProfileResponse ::= {  enableResult catBusy  }  SW=0x9000 or 0x91XX |
| 2 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | S\_LPAd → Euicc | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1,  #PROFILE\_INFO2  }  SW=0x9000 |

Test Sequence #06 Error: Enable Profile by ICCID with refreshFlag set while proactive session is ongoing – catBusy supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the Euicc on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify  ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = ‘03’,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC5 | Do not send FETCH command | | |
| IC6 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF2,  NO\_PARAM,  TRUE)) | resp EnableProfileResponse ::= {  enableResult catBusy  }  SW=0x9000 or 0x91XX |
| 2 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1,  #PROFILE\_INFO2  }  SW=0x9000 |

Test Sequence #07 Error: Enable 2nd Profile by an unknown ISD-P AID where one profile is already enabled

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The Operational Profile identified by the ISD-P AID <ISD\_P\_AIDX> is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC5 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC6 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AIDX>,  TRUE)) | #R\_ENABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  }  SW=0x9000 |

Test Sequence #08 Error: Enable 2nd Profile by an unknown ICCID where one profile is already enabled

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to #ICCID\_OP\_PROF1. |
| eUICC | The Operational Profile identified by the ICCID #ICCID\_OP\_PROFX is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC5 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC6 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROFX,  NO\_PARAM,  TRUE)) | #R\_ENABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  }  SW=0x9000 |

Test Sequence #09 Error: Enable 2nd Profile (by ISD-P AID) is not possible when this Operational Profile is in Enabled state at other LSI

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID2>,  TRUE)) | #R\_ENABLE\_PROFILE\_NOT\_DISABLE\_STATE  SW=0x9000 |
| 2 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 3 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_MEPB;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | #R\_ENABLE\_PROFILE\_NOT\_DISABLE\_STATE  SW=0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_MEPB;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |

Test Sequence #10 Error: Enable 2nd Profile (by ICCID) is not possible when this Operational Profile is in Enabled state at other LSI

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to #ICCID\_OP\_PROF1 |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to #ICCID\_OP\_PROF2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF2,  NO\_PARAM,  TRUE)) | #R\_ENABLE\_PROFILE\_NOT\_DISABLE\_STATE  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | #R\_ENABLE\_PROFILE\_NOT\_DISABLE\_STATE  SW=0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF2,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO2\_ENABLED  }  SW=0x9000 |

Test Sequence #11 Error: Enable 2nd Profile by ISD-P AID and “refreshFlag” set while 2 proactive sessions are ongoing – catBusy supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |
| eUICC | The PROFILE\_OPERATIONAL3 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 corresponds to <ISD\_P\_AID3>. |
| eUICC | The Nickname of the PROFILE\_OPERATIONAL3 is equal to #NICKNAME3. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC8 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91YY |
| IC9 | Do not send FETCH command | | |
| IC10 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID3>,  TRUE)) | resp EnableProfileResponse ::= {  enableResult catBusy  }  SW= 9000 |
| 2 | Repeat IC4 | | |
| 3 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 4 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 5 | Repeat IC7 | | |
| 6 | S\_Device → eUICC | FETCH 'YY' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 7 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_MEPB;  #PROFILE\_INFO3  }  SW=0x9000 |
| 9 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 10 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 11 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_MEPB;  #PROFILE\_INFO2\_ENABLED;  #PROFILE\_INFO3  }  SW=0x9000 |

Test Sequence #12 Error: Enable 2nd Profile (by ISD-P AID) is not possible when CAT is not initialized for targetPort

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP(  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_ONE\_LSI\_FOR\_ENABLED\_PROFILE | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC5 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC6 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID2>,  TRUE)) | resp EnableProfileResponse ::= {  enableResult commandError  }  SW=0x9000 |
| 2 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1,  #PROFILE\_INFO2  }  SW=0x9000 |
| 4 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 5 | S\_Device 🡪 eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

### 4.2.22 ES10c (LPA – eUICC): DisableProfile

#### 4.2.22.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 2.4.5
* Section 2.9.1, 2.9.3.3
* Section 3.2.2
* Section 5.7.17

#### 4.2.22.2 Test Cases

##### 4.2.22.2.1 TC\_eUICC\_ES10c.DisableProfile\_Case3

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |

Test Sequence #01 Nominal: Disable Profile by ISD-P AID and “refreshFlag” set when Device supports “UICC Reset”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | No response data is returned  SW=0x91XX |
| 2 | S\_Device → eUICC | FETCH ‘XX’ | REFRESH Command (“UICC Reset”) |
| 3 | Repeat IC1 and IC2 | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)**)** | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 5 | S\_Device → eUICC | [SELECT\_ICCID] | SW=6A82 |

Test Sequence #02 Nominal: Disable Profile by ICCID and “refreshFlag” set when Device supports “UICC Reset”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | No response data is returned  SW=0x91XX |
| 2 | S\_Device → eUICC | FETCH ‘XX’ | REFRESH Command (“UICC Reset”) |
| 3 | Repeat IC1 and IC2 | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 5 | S\_Device → eUICC | [SELECT\_ICCID] | SW=6A82 |

Test Sequence #03 Nominal: Disable Profile by ISD-P AID and “refreshFlag” set when Device supports “eUICC Profile State Change”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | No response data is returned  SW=0x91XX |
| 2 | S\_Device → eUICC | FETCH ‘XX’ | REFRESH Command (“eUICC Profile State changed”) |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | Execute IC1 from step 2 to step 4 | | |
| 5 | Repeat IC2 | | |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)**)** | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 7 | S\_Device → eUICC | [SELECT\_ICCID] | SW=6A82 |

Test Sequence #04 Nominal: Disable Profile by ICCID and “refreshFlag” set when Device supports “eUICC Profile State Change”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | No response data is returned  SW=0x91XX |
| 2 | S\_Device → eUICC | FETCH ‘XX’ | REFRESH Command (“eUICC Profile State changed”) |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | Execute IC1 from step 2 to step 4 | | |
| 5 | Repeat IC2 | | |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 7 | S\_Device → eUICC | [SELECT\_ICCID] | SW=6A82 |

Test Sequence #05 Nominal: Disable Profile by ISD-P AID and “refreshFlag” no set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  FALSE)) | No response data is returned  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)**)** | response ProfileInfoListResponse::= profileInfoListOk : { #PROFILE\_INFO1\_DISABLED    }  SW=0x9000 |
| 3 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #06 Nominal: Disable Profile by ICCID and “refreshFlag” no set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  FALSE)) | No response data is returned  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 3 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #07 Nominal: Disable Profile by ICCID with refreshFLag set while proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC4 | Do not send FETCH command | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | No response data is returned  SW=0x91YY |
| 2 | S\_Device 🡪eUICC | FETCH ‘YY’ | REFRESH Command (“UICC Reset”) |
| 3 | Repeat IC1 and IC2 | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 5 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #08 Nominal: Disable Profile by ICCID with refreshFLag not set while proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC4 | Do not send FETCH command | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  FALSE)) | No response data is returned  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 3 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #09 Nominal: Disable Profile by ICCID with refreshFLag set while proactive session is ongoing with Terminal Response outstanding – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| 1 | S\_Device 🡪eUICC | FETCH ‘XX’ | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | No response data is returned  SW=0x9000 |
| 3 | S\_Device 🡪 eUICC | TERMINAL RESPONSE | SW=0x91YY |
| 4 | S\_Device 🡪eUICC | FETCH ‘YY’ | REFRESH Command (“UICC Reset”) |
| 5 | Repeat IC1 and IC2 | | |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 7 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #10 Nominal: Disable Profile by ICCID with refreshFLag not set while proactive session is ongoing with Terminal Response outstanding – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| 1 | S\_Device 🡪eUICC | FETCH ‘XX’ | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  FALSE)) | No response data is returned  SW=0x9000 |
| 3 | S\_Device 🡪eUICC | TERMINAL RESPONSE | SW= any value except 91XX |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 5 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x6A82 |

##### 4.2.22.2.2 TC\_eUICC\_ES10c.DisableProfile\_ErrorCases\_Case3

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |

Test Sequence #01 Error: Disable Profile by an unknown ISD-P AID

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The Operational Profile identified by the ISD-P AID <ISD\_P\_AIDX> is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AIDX>,  TRUE)) | SW=0x6A82 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #02 Error: Disable Profile by an unknown ICCID

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The Operational Profile identified by the ICCID #ICCID\_OP\_PROFX is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROFX,  NO\_PARAM,  TRUE)) | SW=0x6A82 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #03 Error: Disable Profile (by ISD-P AID) is not possible when this Operational Profile is in Disabled state

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | SW=0x6985 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)**)** | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #04 Error: Disable Profile (by ICCID) is not possible when this Operational Profile is in Disabled state

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | SW=0x6985 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #05 Error: Disable Profile (by ISD-P AID) not possible when PPR1 is set

The purpose of this test is to ensure that it is NOT possible to disable an Operational Profile4 with the Policy Rule “Disabling of this Profile is not allowed”.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is not loaded.  (this condition overrides the general initial condition defined in this test case) |
| eUICC | The PROFILE\_OPERATIONAL4 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL4 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL4 corresponds to <ISD\_P\_AID4>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID4>,  TRUE)) | SW=0x6985 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID4>)**)** | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO4\_ENABLED  }  SW=0x9000 |

Test Sequence #06 Error: Disable Profile (by ICCID) not possible when PPR1 is set

The purpose of this test is to ensure that it is NOT possible to disable an Operational Profile4 with the Policy Rule “Disabling of this Profile is not allowed”.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is not loaded.  (this condition overrides the general initial condition defined in this test case) |
| eUICC | The PROFILE\_OPERATIONAL4 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL4 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF4,  NO\_PARAM,  TRUE)) | SW=0x6985 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF4,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO4\_ENABLED  }  SW=0x9000 |

Test Sequence #07 Error: Disable Profile by ISDP-AID without refreshFlag while proactive session is ongoing – catBusy supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC4 | Do not send FETCH command | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  FALSE)) | SW=0x9300 |
| 2 | S\_Device → eUICC | FETCH ‘XX’ | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)**)** | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #08 Error: Disable Profile by ICCID with refreshFlag set while proactive session is ongoing – catBusy supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC4 | Do not send FETCH command | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | SW=0x9300 |
| 2 | S\_Device → eUICC | FETCH ‘XX’ | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

##### 4.2.22.2.3 TC\_eUICC\_ES10c.DisableProfile\_Case4

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |

Test Sequence #01 Nominal: Disable Profile by ISD-P AID and “refreshFlag” set when Device supports “UICC Reset”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | S\_Device → eUICC | FETCH ‘XX’ | REFRESH Command (“UICC Reset”) |
| 3 | Repeat IC1 and IC2 | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)**)** | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 5 | S\_Device → eUICC | [SELECT\_ICCID] | SW=6A82 |

Test Sequence #02 Nominal: Disable Profile by ICCID and “refreshFlag” set when Device supports “UICC Reset”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | S\_Device → eUICC | FETCH ‘XX’ | REFRESH Command (“UICC Reset”) |
| 3 | Repeat IC1 and IC2 | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 5 | S\_Device → eUICC | [SELECT\_ICCID] | SW=6A82 |

Test Sequence #03 Nominal: Disable Profile by ISD-P AID and “refreshFlag” set when Device supports “eUICC Profile State Change”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | S\_Device → eUICC | FETCH ‘XX’ | REFRESH Command (“eUICC Profile State changed”) |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | Execute IC1 from step 2 to step 4 | | |
| 5 | Repeat IC2 | | |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)**)** | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 7 | S\_Device → eUICC | [SELECT\_ICCID] | SW=6A82 |

Test Sequence #04 Nominal: Disable Profile by ICCID and “refreshFlag” set when Device supports “eUICC Profile State Change”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | S\_Device → eUICC | FETCH ‘XX’ | REFRESH Command (“eUICC Profile State changed”) |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | Execute IC1 from step 2 to step 4 | | |
| 5 | Repeat IC2 | | |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 7 | S\_Device → eUICC | [SELECT\_ICCID] | SW=6A82 |

Test Sequence #05 Nominal: Disable Profile by ISD-P AID and “refreshFlag” no set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  FALSE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)**)** | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED    }  SW=0x9000 |
| 3 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #06 Nominal: Disable Profile by ICCID and “refreshFlag” no set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  FALSE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 3 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #07 Nominal: Disable Profile by ISD-P AID and “refreshFlag” set while proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC4 | Do not send FETCH command | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | resp DisableProfileResponse ::= {  DisableResult ok  }  SW=0x91YY |
| 2 | S\_Device 🡪eUICC | FETCH ‘YY’ | REFRESH Command (“UICC Reset”) |
| 3 | Repeat IC1 and IC2 | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 5 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #08 Nominal: Disable Profile by ISD-P AID and “refreshFlag” not set while proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC4 | Do not send FETCH command | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  FALSE)) | resp DisableProfileResponse ::= {  DisableResult ok  }  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 3 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #09 Nominal: Disable Profile by ISD-P AID and “refreshFlag” set while proactive session is ongoing with Terminal Response outstanding – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| 1 | S\_Device 🡪eUICC | FETCH ‘XX’ | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | resp DisableProfileResponse ::= {  DisableResult ok  }  SW=0x9000 |
| 3 | S\_Device 🡪 eUICC | TERMINAL RESPONSE | SW=0x91YY |
| 4 | S\_Device 🡪eUICC | FETCH ‘YY’ | REFRESH Command (“UICC Reset”) |
| 5 | Repeat IC1 and IC2 | | |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 7 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #10 Nominal: Disable Profile by ISD-P AID and “refreshFlag” not set while proactive session is ongoing with Terminal Response outstanding – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| 1 | S\_Device 🡪eUICC | FETCH ‘XX’ | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  FALSE)) | resp DisableProfileResponse ::= {  DisableResult ok  }  SW=0x9000 |
| 3 | S\_Device 🡪 eUICC | TERMINAL RESPONSE | SW= any value except 91XX |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 5 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x6A82 |

##### 4.2.22.2.4 TC\_eUICC\_ES10c.DisableProfile\_ErrorCases\_Case4

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |

Test Sequence #01 Error: Disable Profile by an unknown ISD-P AID

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The Operational Profile identified by the ISD-P AID <ISD\_P\_AIDX> is not loaded. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step | Direction | | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AIDX>,  TRUE)) | | #R\_DISABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #02 Error: Disable Profile by an unknown ICCID

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The Operational Profile identified by the ICCID #ICCID\_OP\_PROFX is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROFX,  NO\_PARAM,  TRUE)) | #R\_DISABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #03 Error: Disable Profile (by ISD-P AID) is not possible when this Operational Profile is in Disabled state

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | #R\_DISABLE\_PROFILE\_NOT\_ENABLE\_STATE  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)**)** | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #04 Error: Disable Profile (by ICCID) is not possible when this Operational Profile is in Disabled state

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | #R\_DISABLE\_PROFILE\_NOT\_ENABLE\_STATE  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #05 Error: Disable Profile (by ISD-P AID) not possible when PPR1 is set

The purpose of this test is to ensure that it is NOT possible to disable an Operational Profile with the Policy Rule “Disabling of this Profile is not allowed”.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is not loaded.  (this condition overrides the general initial condition defined in this test case) |
| eUICC | The PROFILE\_OPERATIONAL4 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL4 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL4 corresponds to <ISD\_P\_AID4>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID4>,  TRUE)) | #R\_DISABLE\_PROFILE\_DISALLOWEDbyPOLICY  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID4>)**)** | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO4\_ENABLED  }  SW=0x9000 |

Test Sequence #06 Error: Disable Profile (by ICCID) not possible when PPR1 is set

The purpose of this test is to ensure that it is NOT possible to disable an Operational Profile4 with the Policy Rule “Disabling of this Profile is not allowed”.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is not loaded.  (this condition overrides the general initial condition defined in this test case) |
| eUICC | The PROFILE\_OPERATIONAL4 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL4 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF4,  NO\_PARAM,  TRUE)) | #R\_DISABLE\_PROFILE\_DISALLOWEDbyPOLICY  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF4,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO4\_ENABLED  }  SW=0x9000 |

Test Sequence #07 Error: Disable Profile by ISD-P AID without refreshFlag while proactive session is ongoing – catBusy supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC4 | Do not send FETCH command | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  FALSE)) | resp DisableProfileResponse ::= {  disableResult catBusy  }  SW=0x9000 or 0x91XX |
| 2 | S\_Device → eUICC | FETCH ‘XX’ | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #08 Error: DisableProfile by ICCID with refreshFlag set while proactive session is ongoing – catBusy supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC4 | Do not send FETCH command | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | resp DisableProfileResponse ::= {  disableResult catBusy  }  SW=0x9000 or 0x91XX |
| 2 | S\_Device → eUICC | FETCH ‘XX’ | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

##### 4.2.22.2.5 TC\_eUICC\_ES10c.DisableProfile\_Case4\_MEPA1

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | eUICC in MEP mode |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |

Test Sequence #01 Nominal: Disable Profile by ISD-P AID and “refreshFlag” set when Device supports “UICC Reset”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE (  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  1,  “UICC Reset”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_FIRST\_PROFILE | | |
| 4 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 5 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 6 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=6A82 |

Test Sequence #02 Nominal: Disable Profile by ICCID and “refreshFlag” set when Device supports “UICC Reset”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  0,  “UICC Reset”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_FIRST\_PROFILE | | |
| 4 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 5 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 6 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=6A82 |

Test Sequence #03 Nominal: Disable Profile by ISD-P AID and “refreshFlag” set when Device supports “eUICC Profile State Change”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  1,  “eUICC Profile State Change”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP\_EN\_DS\_FIRST\_PROFILE | | |
| 4 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 5 | proc\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 6 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=6A82 |

Test Sequence #04 Nominal: Disable Profile by ICCID and “refreshFlag” set when Device supports “eUICC Profile State Change”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE (  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  0,  “eUICC Profile State Change”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP\_EN\_DS\_FIRST\_PROFILE | | |
| 4 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 5 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 6 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=6A82 |

Test Sequence #05 Nominal: Disable Profile by ISD-P AID and “refreshFlag” not set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  FALSE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x9000 |
| 2 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED    }  SW=0x9000 |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 4 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #06 Nominal: Disable Profile by ICCID and “refreshFlag” not set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE (  #ICCID\_OP\_PROF1,  NO\_PARAM,  FALSE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x9000 |
| 2 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 4 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #07 Nominal: Disable Profile by ISD-P AID and “refreshFlag” set while proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC8 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | resp DisableProfileResponse ::= {  DisableResult ok  }  SW=0x91YY |
| 2 | S\_Device → eUICC | PROC\_MEP\_REFRESH\_EN\_DS(  0,  “UICC Reset”) | Verify <LSI\_COMMAND\_ACTION> = “Proactive session request” and <LSI\_NUMBER> = 1  REFRESH Command (“UICC Reset”) |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_FIRST\_PROFILE | | |
| 4 | PROC\_ MEP\_LSI\_MULTIPLEXING(0) | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 6 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 7 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #08 Nominal: Disable Profile by ISD-P AID and “refreshFlag” not set while proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = #01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC8 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  FALSE)) | resp DisableProfileResponse ::= {  DisableResult ok  }  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 4 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #09 Nominal: Disable 2nd Profile by ISD-P AID and “refreshFlag” set when Device supports “UICC Reset”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 2. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID2>,  TRUE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  2,  “UICC Reset”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2;  }  SW=0x9000 |
| 5 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| 6 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #10 Nominal: Disable 2nd Profile by ICCID and “refreshFlag” set when Device supports “UICC Reset”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to #ICCID\_OP\_PROF1 |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 2. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to #ICCID\_OP\_PROF2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF2,  NO\_PARAM,  TRUE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  2,  “UICC Reset”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2;  }  SW=0x9000 |
| 5 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| 6 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #11 Nominal: Disable 2nd Profile by ISD-P AID and “refreshFlag” set when Device supports “eUICC Profile State Change”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 2. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID2>,  TRUE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  0,  “eUICC Profile State Change”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2;  }  SW=0x9000 |
| 5 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| 6 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #12 Nominal: Disable 2nd Profile by ICCID and “refreshFlag” set when Device supports “eUICC Profile State Change”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to #ICCID\_OP\_PROF1 |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 2. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to #ICCID\_OP\_PROF2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF2,  NO\_PARAM,  TRUE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  2,  “eUICC Profile State Change”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2;  }  SW=0x9000 |
| 5 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| 6 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #13 Nominal: Disable 2nd Profile by ISD-P AID and “refreshFlag” not set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 2. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID2>,  FALSE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x9000 |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  #PROFILE\_INFO2  }  SW=0x9000 |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| 4 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #14 Nominal: Disable 2nd Profile by ICCID and “refreshFlag” not set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to #ICCID\_OP\_PROF1 |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 2. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to #ICCID\_OP\_PROF2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF2,  NO\_PARAM,  FALSE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x9000 |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  #PROFILE\_INFO2  }  SW=0x9000 |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| 4 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #15 Nominal: Disable 2nd Profile by ISD-P AID and “refreshFlag” set while proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 2. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC8 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID2>,  TRUE)) | resp disableProfileResponse ::= {  disableResult ok  }  SW=0x91YY |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  2,  “UICC Reset”) | | |
| IC9 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1,  #PROFILE\_INFO2  }  SW=0x9000 |
| 4 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| 5 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #16 Nominal: Disable 2nd Profile by ISD-P AID and “refreshFlag” set while proactive session is ongoing with Terminal Response outstanding – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | eUICC is MEPA1 capable |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 2. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | S\_Device → eUICC | FETCH ‘XX’ | SMS POR received  SCP80 response status code equal to 0x00 – POR OK and waiting for Terminal Response |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC8 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | resp DisableProfileResponse ::= {  disableResult ok  }  SW=0x91YY |
| 2 | S\_Device 🡪eUICC | FETCH ‘YY’ | Verify <LSI\_COMMAND\_ACTION> = “Proactive session request” and <LSI\_NUMBER> = 1  LSI COMMAND (“ProactiveSessionRequest”) |
| 3 | S\_Device 🡪eUICC | MANAGE\_LSI(Select LSI) for <LSI\_NUMBER> | Switch to targetPort |
| 4 | S\_Device 🡪eUICC | STATUS command | SW=0x9000 |
| 5 | S\_Device 🡪eUICC | TERMINAL RESPONSE | SW=0x91ZZ |
| 6 | S\_Device 🡪eUICC | FETCH ‘ZZ’ | REFRESH Command (“UICC Reset”) |
| 7 | S\_Device → eUICC | [SELECT\_MF] | FCP Template present  SW=0x9000 |
| 8 | S\_Device → eUICC | [TERMINAL\_CAPABILITY\_LPAd] | SW=0x9000 |
| 9 | S\_Device → eUICC | [TERMINAL\_PROFILE\_LSI\_COMMAND] | SW=0x9000 |
| 10 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 11 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |
| 12 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 13 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #17 Nominal: Disable 2nd Profile by ISD-P AID and “refreshFlag” set while 2 proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 2. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC8 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91YY |
| IC9 | Do not send FETCH command | | |
| IC10 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC11 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | resp DisableProfileResponse ::= {  disableResult ok  }  SW=0x91ZZ |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  0,  UICC Reset) | | |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| 4 | S\_Device 🡪eUICC | FETCH ‘XX’ | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 5 | S\_Device 🡪eUICC | TERMINAL RESPONSE |  |
| 6 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO (  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |
| 8 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 9 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #18 Nominal: Disable 2nd Profile by ISD-P AID and “refreshFlag” set while 2 proactive session is ongoing with Terminal Response outstanding – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | eUICC is MEPA1 capable |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 2. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | S\_Device → eUICC | FETCH ‘XX’ | SMS POR received  SCP80 response status code equal to 0x00 – POR OK and waiting for Terminal Response |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC8 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91YY |
| IC9 | S\_Device → eUICC | FETCH ‘YY’ | SMS POR received  SCP80 response status code equal to 0x00 – POR OK and waiting for Terminal Response |
| IC10 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC11 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | resp DisableProfileResponse ::= {  disableResult ok  }  SW=0x91ZZ |
| 2 | S\_Device 🡪eUICC | FETCH ‘ZZ’ | Verify <LSI\_COMMAND\_ACTION> = “Proactive session request” and <LSI\_NUMBER> = 1  LSI COMMAND (“ProactiveSessionRequest”) |
| 3 | S\_Device 🡪eUICC | MANAGE\_LSI(Select LSI) for <LSI\_NUMBER> | Switch to targetPort |
| 4 | S\_Device 🡪eUICC | STATUS command | SW=0x9000 |
| 5 | S\_Device 🡪eUICC | TERMINAL RESPONSE | SW=0x91KK |
| 6 | S\_Device 🡪eUICC | FETCH ‘KK’ | REFRESH Command (“UICC Reset”) |
| 7 | S\_Device → eUICC | [SELECT\_MF] | FCP Template present  SW=0x9000 |
| 8 | S\_Device → eUICC | [TERMINAL\_CAPABILITY\_LPAd] | SW=0x9000 |
| 9 | S\_Device → eUICC | [TERMINAL\_PROFILE\_LSI\_COMMAND] | SW=0x9000 |
| 10 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| 11 | S\_Device 🡪eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 12 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 13 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |
| 14 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 15 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #19 Nominal: Disable 2nd Profile (by ISD-P AID) with Refresh ON is performed successfully when CAT is not initialized for targetPort

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 2. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_ONE\_LSI\_FOR\_ENABLED\_PROFILE | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID2>,  TRUE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x9000 |
| 2 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2;  }  SW=0x9000 |
| 4 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| 5 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x6A82 |

##### 4.2.22.2.6 TC\_eUICC\_ES10c.DisableProfile\_ErrorCases\_Case4\_MEPA1

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | eUICC in MEP mode |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |

Test Sequence #01 Error: Disable Profile by an unknown ISD-P AID

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The Operational Profile identified by the ISD-P AID <ISD\_P\_AIDX> is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AIDX>,  TRUE)) | #R\_DISABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #02 Error: Disable Profile by an unknown ICCID

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The Operational Profile identified by the ICCID #ICCID\_OP\_PROFX is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE (  #ICCID\_OP\_PROFX,  NO\_PARAM,  TRUE)) | #R\_DISABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #03 Error: Disable Profile (by ISD-P AID) is not possible when this Operational Profile is in Disabled state

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | #R\_DISABLE\_PROFILE\_NOT\_ENABLE\_STATE  SW=0x9000 |
| 2 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #04 Error: Disable Profile (by ICCID) is not possible when this Operational Profile is in Disabled state

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | #R\_DISABLE\_PROFILE\_NOT\_ENABLE\_STATE  SW=0x9000 |
| 2 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #05 Error: Disable Profile by ISD-P AID without refreshFlag while proactive session is ongoing – catBusy supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC8 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  FALSE)) | resp DisableProfileResponse ::= {  disableResult catBusy  }  SW=0x9000 |
| 2 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 3 | S\_Device 🡪eUICC | FETCH ‘XX’ | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 4 | S\_Device 🡪 eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #06 Error: DisableProfile by ICCID with refreshFlag set while proactive session is ongoing – catBusy supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on Port 1. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC8 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | resp DisableProfileResponse ::= {  disableResult catBusy  }  SW=0x9000 |
| 2 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 3 | S\_Device 🡪eUICC | FETCH ‘XX’ | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 4 | S\_Device 🡪 eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #07 Error: Disable 3rd Profile by an unknown ISD-P AID where two profile is already enabled

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 2. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |
| eUICC | The Operational Profile identified by the ISD-P AID <ISD\_P\_AIDX> is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC5 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AIDX>,  TRUE)) | #R\_DISABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |

Test Sequence #08 Error: Disable 3rd Profile by an unknown ICCID where two profile is already enabled

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to #ICCID\_OP\_PROF1. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 2. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to #ICCID\_OP\_PROF2. |
| eUICC | The Operational Profile identified by the ICCID #ICCID\_OP\_PROFX is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP \(  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROFX,  NO\_PARAM,  TRUE)) | #R\_DISABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |

Test Sequence #09 Error: Disable 3rd Profile (by ISD-P AID) is not possible when this Operational Profile is in Disabled State

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 2. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 corresponds to <ISD\_P\_AID3>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID3>,  TRUE)) | #R\_DISABLE\_PROFILE\_NOT\_ENABLE\_STATE  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |

Test Sequence #10 Error: Disable 3rd Profile (by ICCID) is not possible when this Operational Profile is in Disabled state

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to #ICCID\_OP\_PROF1 |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 2. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to #ICCID\_OP\_PROF2 |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 corresponds to #ICCID\_OP\_PROF3 |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF3,  NO\_PARAM,  TRUE,  <MEP\_MODE>,  1)) | #R\_DISABLE\_PROFILE\_NOT\_ENABLE\_STATE  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  }  SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF2,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |

Test Sequence #11 Error: Disable 3rd Profile by ISD-P AID and “refreshFlag” set while 2 proactive session is ongoing – catBusy supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 2. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC8 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91YY |
| IC9 | Do not send FETCH command | | |
| IC10 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC11 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | resp DisableProfileResponse ::= {  disableResult catBusy  }  SW=0x9000 |
| 2 | Repeat IC7 | | |
| 3 | S\_Device → eUICC | FETCH ‘XX’ | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 4 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 5 | Repeat IC4 | | |
| 6 | S\_Device → eUICC | FETCH ‘YY’ | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 7 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1,  #PROFILE\_INFO2\_ENABLED  }  SW=0x9000 |

##### 4.2.22.2.7 TC\_eUICC\_ES10c.DisableProfile\_Case4\_MEPA2

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | eUICC in MEP mode |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |

Test Sequence #01 Nominal: Disable Profile by ISD-P AID and “refreshFlag” not set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on target port <MEP-A2\_TARGET\_ESIM\_PORT> assigned by the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE (  NO\_PARAM,  <ISD\_P\_AID1>,  FALSE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x9000 |
| 2 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED    }  SW=0x9000 |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(<MEP-A2\_TARGET\_ESIM\_PORT>) | | |
| 4 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #02 Nominal: Disable Profile by ICCID and “refreshFlag” not set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on target port <MEP-A2\_TARGET\_ESIM\_PORT> assigned by the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  FALSE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x9000 |
| 2 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(<MEP-A2\_TARGET\_ESIM\_PORT>) | | |
| 4 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #03 Nominal: Disable Profile by ISD-P AID and “refreshFlag” not set while proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on target port <MEP-A2\_TARGET\_ESIM\_PORT> assigned by the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(<MEP-A2\_TARGET\_ESIM\_PORT>) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC8 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  FALSE)) | resp DisableProfileResponse ::= {  DisableResult ok  }  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(<MEP-A2\_TARGET\_ESIM\_PORT>) | | |
| 4 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #04 Nominal: Disable 2nd Profile by ISD-P AID and “refreshFlag” not set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port X. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port Y. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID2>,  FALSE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x9000 |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  #PROFILE\_INFO2  }  SW=0x9000 |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(Y) | | |
| 4 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #05 Nominal: Disable 2nd Profile by ICCID and “refreshFlag” not set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port X. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to #ICCID\_OP\_PROF1 |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port Y. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to #ICCID\_OP\_PROF2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF2,  NO\_PARAM,  FALSE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x9000 |
| 2 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  #PROFILE\_INFO2  }  SW=0x9000 |
| 4 | PROC\_MEP\_LSI\_MULTIPLEXING(Y) | | |
| 5 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x6A82 |

##### 4.2.22.2.8 TC\_eUICC\_ES10c.DisableProfile\_ErrorCases\_Case4\_MEPA2

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | eUICC in MEP mode |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |

Test Sequence #01 Error: Disable Profile by an unknown ISD-P AID

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The Operational Profile identified by the ISD-P AID <ISD\_P\_AIDX> is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AIDX>,  FALSE)) | #R\_DISABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #02 Error: Disable Profile by an unknown ICCID

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The Operational Profile identified by the ICCID #ICCID\_OP\_PROFX is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE (  #ICCID\_OP\_PROFX,  NO\_PARAM,  FALSE)) | #R\_DISABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #03 Error: Disable Profile (by ISD-P AID) is not possible when this Operational Profile is in Disabled state

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  FALSE)) | #R\_DISABLE\_PROFILE\_NOT\_ENABLE\_STATE  SW=0x9000 |
| 2 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #04 Error: Disable Profile (by ICCID) is not possible when this Operational Profile is in Disabled state

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  FALSE)) | #R\_DISABLE\_PROFILE\_NOT\_ENABLE\_STATE  SW=0x9000 |
| 2 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #05 Error: Disable Profile by ISD-P AID without refreshFlag while proactive session is ongoing – catBusy supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on target port <MEP-A2\_TARGET\_ESIM\_PORT> assigned by the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(<MEP-A2\_TARGET\_ESIM\_PORT>) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC8 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  FALSE)) | resp DisableProfileResponse ::= {  disableResult catBusy  }  SW=0x9000 |
| 2 | PROC\_MEP\_LSI\_MULTIPLEXING(<MEP-A2\_TARGET\_ESIM\_PORT>) | | |
| 3 | S\_Device 🡪eUICC | FETCH ‘XX’ | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 4 | S\_Device 🡪 eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #06 Error: Disable 3rd Profile by an unknown ISD-P AID where two profile is already enabled

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |
| eUICC | The Operational Profile identified by the ISD-P AID <ISD\_P\_AIDX> is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC5 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AIDX>,  FALSE)) | #R\_DISABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |

Test Sequence #07 Error: Disable 3rd Profile by an unknown ICCID where two profile is already enabled

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to #ICCID\_OP\_PROF1. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to #ICCID\_OP\_PROF2. |
| eUICC | The Operational Profile identified by the ICCID #ICCID\_OP\_PROFX is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROFX,  NO\_PARAM,  FALSE)) | #R\_DISABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |

Test Sequence #08 Error: Disable 3rd Profile (by ISD-P AID) is not possible when this Operational Profile is in Disabled State

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 corresponds to <ISD\_P\_AID3>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID3>,  FALSE)) | #R\_DISABLE\_PROFILE\_NOT\_ENABLE\_STATE  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |

Test Sequence #09 Error: Disable 3rd Profile (by ICCID) is not possible when this Operational Profile is in Disabled state

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to #ICCID\_OP\_PROF1 |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to #ICCID\_OP\_PROF2 |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 corresponds to #ICCID\_OP\_PROF3 |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_ENABLE\_PROFILE(  #ICCID\_OP\_PROF3,  NO\_PARAM,  FALSE,  <MEP\_MODE>,  1)) | #R\_DISABLE\_PROFILE\_NOT\_ENABLE\_STATE  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  }  SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF2,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |

Test Sequence #10 Error: Disable 3rd Profile by ISD-P AID and “refreshFlag” not set while 2 proactive session is ongoing – catBusy supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port X. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port Y. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(Y) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(X) | | |
| IC8 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91YY |
| IC9 | Do not send FETCH command | | |
| IC10 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC11 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE (  NO\_PARAM,  <ISD\_P\_AID1>,  FALSE)) | resp DisableProfileResponse ::= {  disableResult catBusy  }  SW=0x9000 |
| 2 | Repeat IC7 | | |
| 3 | S\_Device → eUICC | FETCH ‘XX’ | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 4 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 5 | Repeat IC4 | | |
| 6 | S\_Device → eUICC | FETCH ‘YY’ | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 7 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1,  #PROFILE\_INFO2\_ENABLED  }  SW=0x9000 |

##### 4.2.22.2.9 TC\_eUICC\_ES10c.DisableProfile\_Case4\_MEPB

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | eUICC in MEP mode |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |

Test Sequence #01 Nominal: Disable Profile by ISD-P AID and “refreshFlag” set when Device supports “UICC Reset”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  0,  “UICC Reset”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_FIRST\_PROFILE | | |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 6 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=6A82 |

Test Sequence #02 Nominal: Disable Profile by ICCID and “refreshFlag” set when Device supports “UICC Reset”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE (  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  0,  “UICC Reset”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_FIRST\_PROFILE | | |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd 🡪Euicc | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 6 | S\_Device 🡪 Euicc | [SELECT\_ICCID] | SW=6A82 |

Test Sequence #03 Nominal: Disable Profile by ISD-P AID and “refreshFlag” set when Device supports “Euicc Profile State Change”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| Euicc | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE (  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  0,  “eUICC Profile State Change”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP\_EN\_DS\_FIRST\_PROFILE | | |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 6 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=6A82 |

Test Sequence #04 Nominal: Disable Profile by ICCID and “refreshFlag” set when Device supports “eUICC Profile State Change”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE (  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  0,  “eUICC Profile State Change”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP\_EN\_DS\_FIRST\_PROFILE | | |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 6 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=6A82 |

Test Sequence #05 Nominal: Disable Profile by ISD-P AID and “refreshFlag” not set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  FALSE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x9000 |
| 2 | S\_Device → eUICC | [TERMINAL\_PROFILE\_LSI\_COMMAND] | Toolkit initialization THEN SW=0x9000 |
| 3 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED    }  SW=0x9000 |
| 4 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #06 Nominal: Disable Profile by ICCID and “refreshFlag” not set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICCon Port 0. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  FALSE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x9000 |
| 2 | S\_Device → eUICC | [TERMINAL\_PROFILE\_LSI\_COMMAND] | Toolkit initialization THEN SW=0x9000 |
| 3 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 4 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #07 Nominal: Disable Profile by ISD-P AID and “refreshFlag” set while proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC5 | Do not send FETCH command | | |
| IC6 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | resp DisableProfileResponse ::= {  DisableResult ok  }  SW=0x91YY |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  0,  “UICC Reset”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_FIRST\_PROFILE | | |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 6 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #08 Nominal: Disable Profile by ISD-P AID and “refreshFlag” not set while proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC5 | Do not send FETCH command | | |
| IC6 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  FALSE)) | resp DisableProfileResponse ::= {  DisableResult ok  }  SW=0x9000 |
| 2 | S\_Device → eUICC | [TERMINAL\_PROFILE\_LSI\_COMMAND] | Toolkit initialization THEN SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 4 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #09 Nominal: Disable 2nd Profile by ISD-P AID and “refreshFlag” set when Device supports “UICC Reset”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC5 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC6 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID2>,  TRUE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  0,  “UICC Reset”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_MEPB;  #PROFILE\_INFO2;  }  SW=0x9000 |
| 6 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #10 Nominal: Disable 2nd Profile by ICCID and “refreshFlag” set when Device supports “UICC Reset”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to #ICCID\_OP\_PROF1 |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to #ICCID\_OP\_PROF2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE (  #ICCID\_OP\_PROF2,  NO\_PARAM,  TRUE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  0,  “UICC Reset”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → Euicc | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_MEPB;  #PROFILE\_INFO2;  }  SW=0x9000 |
| 6 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #11 Nominal: Disable 2nd Profile by ISD-P AID and “refreshFlag” set when Device supports “eUICC Profile State Change”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE (  NO\_PARAM,  <ISD\_P\_AID2>,  TRUE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  0,  “eUICC Profile State Change”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_MEPB;  #PROFILE\_INFO2;  }  SW=0x9000 |
| 6 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #12 Nominal: Disable 2nd Profile by ICCID and “refreshFlag” set when Device supports “eUICC Profile State Change”

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to #ICCID\_OP\_PROF1 |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to #ICCID\_OP\_PROF2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE (  #ICCID\_OP\_PROF2,  NO\_PARAM,  TRUE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x91XX |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  0,  “eUICC Profile State Change”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_MEPB;  #PROFILE\_INFO2;  }  SW=0x9000 |
| 6 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #13 Nominal: Disable 2nd Profile by ISD-P AID and “refreshFlag” not set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE (  NO\_PARAM,  <ISD\_P\_AID2>,  FALSE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x9000 |
| 2 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO (  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_MEPB  #PROFILE\_INFO2  }  SW=0x9000 |
| 4 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #14 Nominal: Disable 2nd Profile by ICCID and “refreshFlag” not set

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to #ICCID\_OP\_PROF1 |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to #ICCID\_OP\_PROF2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF2,  NO\_PARAM,  FALSE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x9000 |
| 2 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 3 | S\_LPAd → Euicc | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_MEPB  #PROFILE\_INFO2  }  SW=0x9000 |
| 4 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #15 Nominal: Disable 2nd Profile by ISD-P AID and “refreshFlag” set while proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE (  NO\_PARAM,  <ISD\_P\_AID2>,  TRUE)) | resp disableProfileResponse ::= {  disableResult ok  }  SW=0x91YY |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  1,  “UICC Reset”) | | |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1\_MEPB,  #PROFILE\_INFO2  }  SW=0x9000 |
| 6 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #16 Nominal: Disable 2nd Profile by ISD-P AID and “refreshFlag” set while proactive session is ongoing with Terminal Response outstanding – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | eUICC is MEPA1 capable |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | S\_Device → eUICC | FETCH ‘XX’ | SMS POR received  SCP80 response status code equal to 0x00 – POR OK and waiting for Terminal Response |
| IC7 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE (  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | resp DisableProfileResponse ::= {  disableResult ok  }  SW=0x9000 |
| 2 | S\_Device 🡪eUICC | STATUS command | SW=0x9000 |
| 3 | S\_Device 🡪eUICC | TERMINAL RESPONSE | SW=0x91ZZ |
| 4 | PROC\_MEP\_REFRESH\_EN\_DS(  1,  “UICC Reset”) | | |
| 5 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 6 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <ISD\_P\_AID2>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO2  }  SW=0x9000 |
| 8 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #17 Nominal: Disable 2nd Profile by ISD-P AID and “refreshFlag” set while 2 proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC8 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91YY |
| IC9 | Do not send FETCH command | | |
| IC10 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | resp DisableProfileResponse ::= {  disableResult ok  }  SW=0x91ZZ |
| 2 | PROC\_MEP\_REFRESH\_EN\_DS(  0,  UICC Reset) | | |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 4 | S\_Device 🡪eUICC | FETCH ‘XX’ | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 5 | S\_Device 🡪eUICC | TERMINAL RESPONSE |  |
| 6 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 7 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_FIRST\_PROFILE | | |
| 8 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 10 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #18 Nominal: Disable 2nd Profile by ISD-P AID and “refreshFlag” set while 2 proactive session is ongoing with Terminal Response outstanding – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | eUICC is MEPA1 capable |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | S\_Device → eUICC | FETCH ‘XX’ | SMS POR received  SCP80 response status code equal to 0x00 – POR OK and waiting for Terminal Response |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC8 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91YY |
| IC9 | S\_Device → eUICC | FETCH ‘YY’ | SMS POR received  SCP80 response status code equal to 0x00 – POR OK and waiting for Terminal Response |
| IC10 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE (  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | resp DisableProfileResponse ::= {  disableResult ok  }  SW=0x9000 |
| 2 | S\_Device 🡪eUICC | STATUS command | SW=0x9000 |
| 3 | S\_Device 🡪eUICC | TERMINAL RESPONSE | SW=0x91KK |
| 4 | PROC\_MEP\_REFRESH\_EN\_DS(  1,  “UICC Reset”) | | |
| 5 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_FIRST\_PROFILE | | |
| 6 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 7 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 8 | S\_Device 🡪eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 9 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 10 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED;  #PROFILE\_INFO2\_MEPB;  }  SW=0x9000 |
| 11 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x6A82 |

Test Sequence #19 Nominal: Disable 2nd Profile (by ISD-P AID) with “refreshFlag” set is performed successfully when CAT is not initialized for targetPort

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_ONE\_LSI\_FOR\_ENABLED\_PROFILE | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE (  NO\_PARAM,  <ISD\_P\_AID2>,  TRUE)) | #R\_DISABLE\_PROFILE\_OK  SW=0x9000 |
| 2 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_SECOND\_PROFILE | | |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2;  }  SW=0x9000 |
| 5 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 6 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x6A82 |

##### 4.2.22.2.10 TC\_eUICC\_ES10c.DisableProfile\_ErrorCases\_Case4\_MEPB

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | eUICC in MEP mode |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |

Test Sequence #01 Error: Disable Profile by an unknown ISD-P AID

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The Operational Profile identified by the ISD-P AID <ISD\_P\_AIDX> is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE (  NO\_PARAM,  <ISD\_P\_AIDX>,  TRUE)) | #R\_DISABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #02 Error: Disable Profile by an unknown ICCID

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The Operational Profile identified by the ICCID #ICCID\_OP\_PROFX is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROFX,  NO\_PARAM,  TRUE)) | #R\_DISABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #03 Error: Disable Profile (by ISD-P AID) is not possible when this Operational Profile is in Disabled state

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | #R\_DISABLE\_PROFILE\_NOT\_ENABLE\_STATE  SW=0x9000 |
| 2 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #04 Error: Disable Profile (by ICCID) is not possible when this Operational Profile is in Disabled state

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE (  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | #R\_DISABLE\_PROFILE\_NOT\_ENABLE\_STATE  SW=0x9000 |
| 2 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #05 Error: Disable Profile by ISD-P AID without refreshFlag while proactive session is ongoing – catBusy supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC5 | Do not send FETCH command | | |
| IC6 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE (  NO\_PARAM,  <ISD\_P\_AID1>,  FALSE)) | resp DisableProfileResponse ::= {  disableResult catBusy  }  SW=0x9000 |
| 2 | S\_Device 🡪eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 3 | S\_Device 🡪 eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #06 Error: DisableProfile by ICCID with refreshFlag set while proactive session is ongoing – catBusy supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on Port 0. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC5 | Do not send FETCH command | | |
| IC6 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM,  TRUE)) | resp DisableProfileResponse ::= {  disableResult catBusy  }  SW=0x9000 |
| 2 | S\_Device 🡪eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 3 | S\_Device 🡪 eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #07 Error: Disable 3rd Profile by an unknown ISD-P AID where two profiles are already enabled

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |
| eUICC | The Operational Profile identified by the ISD-P AID <ISD\_P\_AIDX> is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC5 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AIDX>,  TRUE)) | #R\_DISABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_MEPB  }  SW=0x9000 |

Test Sequence #08 Error: Disable 3rd Profile by an unknown ICCID where two profiles are already enabled

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to #ICCID\_OP\_PROF1. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to #ICCID\_OP\_PROF2. |
| eUICC | The Operational Profile identified by the ICCID #ICCID\_OP\_PROFX is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROFX,  NO\_PARAM,  TRUE)) | #R\_DISABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_MEPB  }  SW=0x9000 |

Test Sequence #09 Error: Disable 3rd Profile (by ISD-P AID) is not possible when this Operational Profile is in Disabled State

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 corresponds to <ISD\_P\_AID3>. |
| eUICC | The Nickname of the PROFILE\_OPERATIONAL3 is equal to #NICKNAME3. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID3>,  TRUE)) | #R\_DISABLE\_PROFILE\_NOT\_ENABLE\_STATE  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_MEPB;  #PROFILE\_INFO3  }  SW=0x9000 |

Test Sequence #10 Error: Disable 3rd Profile (by ICCID) is not possible when this Operational Profile is in Disabled state

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to #ICCID\_OP\_PROF1 |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to #ICCID\_OP\_PROF2 |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 corresponds to #ICCID\_OP\_PROF3 |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  #ICCID\_OP\_PROF3,  NO\_PARAM,  TRUE)) | #R\_DISABLE\_PROFILE\_NOT\_ENABLE\_STATE  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF2,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO2\_MEPB  }  SW=0x9000 |

Test Sequence #11 Error: Disable 3rd Profile by ISD-P AID and “refreshFlag” set while 2 proactive sessions are ongoing – catBusy supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030201”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC8 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91YY |
| IC9 | Do not send FETCH command | | |
| IC10 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DISABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID2>,  TRUE)) | resp DisableProfileResponse ::= {  disableResult catBusy  }  SW=0x9000 |
| 2 | Repeat IC4 | | |
| 3 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 4 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 5 | Repeat IC7 | | |
| 6 | S\_Device → eUICC | FETCH 'YY' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 7 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA( MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_MEPB,  #PROFILE\_INFO2\_ENABLED  }  SW=0x9000 |

### 4.2.23 ES10c (LPA -- eUICC): DeleteProfile

#### 4.2.23.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 2.4.4
* Section 2.9.1, 2.9.3.3
* Section 3.2.3
* Section 5.7.15, 5.7.18

#### 4.2.23.2 Test Cases

##### 4.2.23.2.1 TC\_eUICC\_ES10c.DeleteProfile\_Case3

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |

Test Sequence #01 Nominal: Delete Profile by ISD-P AID

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_DELETE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>) | No response data is returned  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)**)** | response ProfileInfoListResponse::= profileInfoListOk: {  }  SW=0x9000 |

Test Sequence #02 Nominal: Delete Profile by ICCID

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_DELETE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM) | No response data is returned  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk: {  }  SW=0x9000 |

##### 4.2.23.2.2 TC\_eUICC\_ES10c.DeleteProfile\_ErrorCases\_Case3

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |

Test Sequence #01 Error: Delete Profile not possible with unknown ISD-P AID

The purpose of this test is to ensure that it is NOT possible to delete an Operational Profile with an unknown ISD-P AID.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The Operational Profile identified by the ISD-P AID <ISD\_P\_AIDX> is not loaded. |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_DELETE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AIDX>) | SW=0x6A82 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk: {  #PROFILE\_INFO1\_DISABLED,  #PROFILE\_INFO2\_ENABLED  }  SW=0x9000 |

Test Sequence #02 Error: Delete Profile not possible with unknown ICCID

The purpose of this test is to ensure that it is NOT possible to delete an Operational Profile with an unknown ICCID.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The Operational Profile identified by the ICCID #ICCID\_OP\_PROFX is not loaded. |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_DELETE\_PROFILE(  #ICCID\_OP\_PROFX,  NO\_PARAM**)** | SW=0x6A82 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk: {  #PROFILE\_INFO1\_DISABLED,  #PROFILE\_INFO2\_ENABLED  }  SW=0x9000 |

Test Sequence #03 Error: Delete Profile (by ISD-P AID) not possible when this Operational Profile is in Enabled state

The purpose of this test is to ensure that it is NOT possible to delete an Operational Profile in Enabled state.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_DELETE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID2>) | SW=0x6985 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA ( #GET\_PROFILES\_INFO\_ALL) | profileInfoListOk: {  #PROFILE\_INFO1\_DISABLED,  #PROFILE\_INFO2\_ENABLED  }  SW=0x9000 |

Test Sequence #04 Error: Delete Profile (by ICCID) not possible when this Operational Profile is in Enabled state

The purpose of this test is to ensure that it is NOT possible to delete an Operational Profile in Enabled state.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_DELETE\_PROFILE(  #ICCID\_OP\_PROF2,  NO\_PARAM**)** | SW=0x6985 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk: {  #PROFILE\_INFO1\_DISABLED,  #PROFILE\_INFO2\_ENABLED  }  SW=0x9000 |

Test Sequence #05 Error: Delete Profile (by ISD-P AID) not possible when PPR2 is set

The purpose of this test is to ensure that it is NOT possible to delete an Operational Profile with the Policy Rule “Deletion of this Profile is not allowed”.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL3 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 corresponds to <ISD\_P\_AID3>. |
| eUICC | The Nickname of the PROFILE\_OPERATIONAL3 is equal to #NICKNAME3. |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_DELETE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID3>) | SW=0x6985 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk: {  #PROFILE\_INFO1\_DISABLED,  #PROFILE\_INFO2\_ENABLED,  #PROFILE\_INFO3  }  SW=0x9000 |

Test Sequence #06 Error: Delete Profile (by ICCID) not possible when PPR2 is set

The purpose of this test is to ensure that it is NOT possible to delete an Operational Profile with the Policy Rule “Deletion of this Profile is not allowed”.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL3 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |
| eUICC | The Nickname of the PROFILE\_OPERATIONAL3 is equal to #NICKNAME3. |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_DELETE\_PROFILE(  #ICCID\_OP\_PROF3,  NO\_PARAM) | SW=0x6985 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk: {  #PROFILE\_INFO1\_DISABLED,  #PROFILE\_INFO2\_ENABLED,  #PROFILE\_INFO3  }  SW=0x9000 |

##### 4.2.23.2.3 TC\_eUICC\_ES10c.DeleteProfile\_Case4

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |

Test Sequence #01 Nominal: Delete Profile by ISD-P AID

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DELETE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>) | #R\_DELETE\_PROFILE\_OK  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  <ISD\_P\_AID1>)**)** | response ProfileInfoListResponse::= profileInfoListOk: {  }  SW=0x9000 |

Test Sequence #02 Nominal: Delete Profile by ICCID

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DELETE\_PROFILE(  #ICCID\_OP\_PROF1,  NO\_PARAM) | #R\_DELETE\_PROFILE\_OK  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk: {  }  SW=0x9000 |

##### 4.2.23.2.4 TC\_eUICC\_ES10c.DeleteProfile\_ErrorCases\_Case4

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |

Test Sequence #01 Error: Delete Profile not possible with unknown ISD-P AID

The purpose of this test is to ensure that it is NOT possible to delete an Operational Profile with an unknown ISD-P AID.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | A Operational Profile identified by the ISD-P AID <ISD\_P\_AIDX> is not loaded. |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DELETE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AIDX>) | #R\_DELETE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk: {  #PROFILE\_INFO1\_DISABLED,  #PROFILE\_INFO2\_ENABLED  }  SW=0x9000 |

Test Sequence #02 Error: Delete Profile not possible with unknown ICCID

The purpose of this test is to ensure that it is NOT possible to delete an Operational with an ICCID unknown.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The Operational Profile identified by the ICCID #ICCID\_OP\_PROFX is not loaded. |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DELETE\_PROFILE(  #ICCID\_OP\_PROFX,  NO\_PARAM**)** | #R\_DELETE\_PROFILE\_ICCID\_ISDP\_NOTFOUND  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk: {  #PROFILE\_INFO1\_DISABLED,  #PROFILE\_INFO2\_ENABLED  }  SW=0x9000 |

Test Sequence #03 Error: Delete Profile (by ISD-P AID) not possible when this Operational Profile is in Enabled state

The purpose of this test is to ensure that it is NOT possible to delete an Operational Profile in Enabled state.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 corresponds to <ISD\_P\_AID2>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DELETE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID2>) | #R\_DELETE\_PROFILE\_NOTDISABLESTATE  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk: {  #PROFILE\_INFO1\_DISABLED,  #PROFILE\_INFO2\_ENABLED  }  SW=0x9000 |

Test Sequence #04 Error: Delete Profile (by ICCID) not possible when this Operational Profile is in Enabled state

The purpose of this test is to ensure that it is NOT possible to delete an Operational Profile in Enabled state.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DELETE\_PROFILE(  #ICCID\_OP\_PROF2,  NO\_PARAM**)** | #R\_DELETE\_PROFILE\_NOTDISABLESTATE  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk: {  #PROFILE\_INFO1\_DISABLED,  #PROFILE\_INFO2\_ENABLED  }  SW=0x9000 |

Test Sequence #05 Error: Delete Profile (by ISD-P AID) not possible when PPR2 is set

The purpose of this test is to ensure that it is NOT possible to delete an Operational Profile with the Policy Rule “Deletion of this Profile is not allowed”.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL3 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 corresponds to <ISD\_P\_AID3>. |
| eUICC | The Nickname of the PROFILE\_OPERATIONAL3 is equal to #NICKNAME3. |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DELETE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID3>) | #R\_DELETE\_PROFILE\_DISALLOWEDBYPOLICY  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk: {  #PROFILE\_INFO1\_DISABLED,  #PROFILE\_INFO2\_ENABLED, #PROFILE\_INFO3  }  SW=0x9000 |

Test Sequence #06 Error: Delete Profile (by ICCID) not possible when PPR2 is set

The purpose of this test is to ensure that it is NOT possible to delete an Operational Profile with the Policy Rule “Deletion of this Profile is not allowed”.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL3 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |
| eUICC | The Nickname of the PROFILE\_OPERATIONAL3 is equal to #NICKNAME3. |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_DELETE\_PROFILE(  #ICCID\_OP\_PROF3,  NO\_PARAM) | #R\_DELETE\_PROFILE\_DISALLOWEDBYPOLICY  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk: {  #PROFILE\_INFO1\_DISABLED,  #PROFILE\_INFO2\_ENABLED,  #PROFILE\_INFO3  }  SW=0x9000 |

### 4.2.24 ES10c (LPA -- eUICC): eUICCMemoryReset

#### 4.2.24.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 2.4.4
* Section 2.9.2
* Section 3.0.1
* Section 3.3.2
* Section 3.5
* Section 5.7.8, 5.7.18, 5.7.19

#### 4.2.24.2 Test Cases

##### 4.2.24.2.1 TC\_eUICC\_ES10c.eUICCMemoryReset

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |
| eUICC | The Default SM-DP+ Address #TEST\_DP\_ADDRESS1 has been set on the ISD-R. |

Test Sequence #01 Nominal: Reset All Operational Profiles (without Enabled Profile)

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |
| eUICC | The Nickname of the PROFILE\_OPERATIONAL3 is equal to #NICKNAME3. |
| eUICC | No Notification is stored in the eUICC's Pending Notifications List. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_EUICC\_INFO2) | Retrieve free non-volatile memory value (tag 0x82) from <EXT\_CARD\_RESOURCE> in EUICCInfo2 as <FREE\_MEM\_OP\_PROF\_INSTALLED> |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk: {  }  SW=0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DE1 SW = 0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_EUICC\_INFO2) | Retrieve free non-volatile memory value (tag 0x82) from <EXT\_CARD\_RESOURCE> in EUICCInfo2 as <FREE\_MEMORY\_NO\_PROFILE>  Verify that <FREE\_MEM\_OP\_PROF\_INSTALLED> is lower than <FREE\_MEMORY\_NO\_PROFILE> |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_EUICC\_CONFIGURED\_DATA) | #R\_ES10a\_GECA\_DS\_DP\_1  SW = 0x9000 |

Test Sequence #02 Nominal: Reset All Operational Profiles (with Enabled Profile), SEP

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | No Notification is stored in the eUICC's Pending Notifications List. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x91XX |
| 2 | S\_Device → eUICC | FETCH 'XX' | REFRESH Command (“UICC Reset”) |
| 3 | Repeat IC1 and IC2 | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DE1 SW = 0x9000  Note : A Disable Notification for PROFILE\_OPERATIONAL1 MAY be also present in the response |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_RAT) | #R\_DEFAULT\_RAT  SW = 0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk: {  }  SW=0x9000 |

Test Sequence #03 Nominal: Reset the Default SM-DP+ Address only

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |
| eUICC | The Nickname of the PROFILE\_OPERATIONAL3 is equal to #NICKNAME3. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_DEF\_SMDPADDRESS) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk: {  #PROFILE\_INFO1\_DISABLED,  #PROFILE\_INFO3  }  SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_EUICC\_CONFIGURED\_DATA) | #R\_ES10a\_GECA\_DS  SW = 0x9000 |

Test Sequence #04 Nominal: Reset All Operational Profiles and the Default SM-DP+ Address

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk: {  }  SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_EUICC\_CONFIGURED\_DATA) | #R\_ES10a\_GECA\_DS  SW = 0x9000 |

Test Sequence #05 Nominal: eUICC Memory Reset, one Operational Profile Enabled, proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | No Notification is stored in the eUICC's Pending Notifications List. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC4 | Do not send FETCH command | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x91YY |
| 2 | S\_Device 🡪eUICC | FETCH 'YY' | REFRESH Command (“UICC Reset”) |
| 3 | Repeat IC1 and IC2 | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DE1 SW = 0x9000  NOTE : A Disable Notification for PROFILE\_OPERATIONAL1 MAY be also present in the response. |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_RAT) | #R\_DEFAULT\_RAT  SW = 0x9000 |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  }  SW=0x9000 |

Test Sequence #06 Nominal: eUICC Memory Reset (with Enabled Profile) while proactive session is ongoing with Terminal Response outstanding – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |
| eUICC | No Notification is stored in the eUICC's Pending Notifications List. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| 1 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x9000 |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x91YY |
| 4 | S\_Device 🡪eUICC | FETCH 'YY' | REFRESH Command (“UICC Reset”) |
| 5 | Repeat IC1 and IC2 | | |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DE1 SW = 0x9000  NOTE : A Disable Notification for PROFILE\_OPERATIONAL1 MAY be also present in the response. |
| **7** | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_RAT) | #R\_DEFAULT\_RAT  SW = 0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  }  SW=0x9000 |

##### 4.2.24.2.2 TC\_eUICC\_ES10c.eUICCMemoryReset\_ErrorCases

Test Sequence #01 Error: eUICC Memory Reset while proactive session is ongoing – catBusy supported

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC4 | Do not send FETCH command | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | resp EuiccMemoryResetResponse::= {  resetResult catBusy  }  SW=0x9000 or 0x91XX |
| 2 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1,  #PROFILE\_INFO2  }  SW=0x9000 |

Test Sequence #02 Error: Nothing to delete

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | No Profile is loaded on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | resp EuiccMemoryResetResponse::= {  resetResult nothingToDelete  }  SW=0x9000 |

##### 4.2.24.2.3 TC\_eUICC\_ES10c.eUICCMemoryReset\_MEPA1

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | eUICC is MEP capable |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |
| eUICC | The Default SM-DP+ Address #TEST\_DP\_ADDRESS1 has been set on the ISD-R. |

Test Sequence #01 Nominal: Reset All Operational Profiles (without Enabled Profile)

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |
| eUICC | No Notification is stored in the eUICC's Pending Notifications List. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_EUICC\_INFO2) | Retrieve free non-volatile memory value (tag 0x82) from <EXT\_CARD\_RESOURCE> in EUICCInfo2 as <FREE\_MEM\_OP\_PROF\_INSTALLED> |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk: {  }  SW=0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DE1\_DE2 SW = 0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_EUICC\_INFO2) | Retrieve free non-volatile memory value (tag 0x82) from <EXT\_CARD\_RESOURCE> in EUICCInfo2 as <FREE\_MEMORY\_NO\_PROFILE>  Verify that <FREE\_MEM\_OP\_PROF\_INSTALLED> is lower than <FREE\_MEMORY\_NO\_PROFILE> |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_EUICC\_CONFIGURED\_DATA) | #R\_ES10a\_GECA\_DS\_DP\_1  SW = 0x9000 |

Test Sequence #02 Nominal: Reset the Default SM-DP+ Address only

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |
| eUICC | The Nickname of the PROFILE\_OPERATIONAL3 is equal to #NICKNAME3. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_DEF\_SMDPADDRESS) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk: {  #PROFILE\_INFO1\_DISABLED,  #PROFILE\_INFO3  }  SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_EUICC\_CONFIGURED\_DATA) | #R\_ES10a\_GECA\_DS  SW = 0x9000 |

Test Sequence #03 Nominal: Reset All Operational Profiles and the Default SM-DP+ Address

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk: {  }  SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_EUICC\_CONFIGURED\_DATA) | #R\_ES10a\_GECA\_DS  SW = 0x9000 |

Test Sequence #04 Nominal: Reset All Operational Profiles (with 2 Enabled Profile)

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 2. |
| eUICC | No Notification is stored in the eUICC’s Pending Notifications List. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x91XX |
| 2 | S\_Device 🡪eUICC | FETCH ‘XX’ | LSI COMMAND (“UICC Platform Reset”) |
| 3 | Repeat IC1 to IC4 | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DE1\_DE2 SW = 0x9000  NOTE : A Disable Notification for PROFILE\_OPERATIONAL1 and PROFILE\_OPERATIONAL2 MAY be also present in the response. |
| 5 | S\_LPAd 🡪 eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  }  SW=0x9000 |

Test Sequence #05 Nominal: Reset the Default SM-DP+ Address only (with 2 Enabled Profile)

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL3 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 is Enabled on the eUICC on Port 2. |
| eUICC | The Nickname of the PROFILE\_OPERATIONAL3 is equal to #NICKNAME3. |
| eUICC | No Notification is stored in the eUICC's Pending Notifications List. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_DEF\_SMDPADDRESS) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x9000 |
| 2 | S\_LPAd 🡪 eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO3\_ENABLED;  }  SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_EUICC\_CONFIGURED\_DATA) | #R\_ES10a\_GECA\_DS  SW = 0x9000 |

Test Sequence #06 Nominal: eUICC Memory Reset, 1 Operational Profile Enabled, proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions | |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | No Notification is stored in the eUICC's Pending Notifications List. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC8 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x91YY |
| 2 | S\_Device 🡪eUICC | FETCH 'YY' | LSI COMMAND (“UICC Platform Reset”) |
| 3 | Repeat IC1 to IC3 | | |
| 4 | Repeat IC8 | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DE1 SW = 0x9000  NOTE : A Disable Notification for PROFILE\_OPERATIONAL1 MAY be also present in the response. |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  }  SW=0x9000 |

Test Sequence #07 Nominal: eUICC Memory Reset, 2 Operational Profile Enabled, 2 proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions | |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 2. |
| eUICC | No Notification is stored in the eUICC's Pending Notifications List. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| IC8 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91YY |
| IC9 | Do not send FETCH command | | |
| IC10 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC11 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x91ZZ |
| 2 | S\_Device 🡪eUICC | FETCH 'ZZ' | LSI COMMAND (“UICC Platform Reset”) |
| 3 | Repeat IC1 to IC3 | | |
| 4 | Repeat IC11 | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DE1\_DE2 SW = 0x9000  NOTE : A Disable Notification for PROFILE\_OPERATIONAL1 and PROFILE\_OPERATIONAL2 MAY be also present in the response. |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  }  SW=0x9000 |

Test Sequence #08 Nominal: eUICC Memory Reset (with Enabled Profile) while proactive session is ongoing with Terminal Response outstanding – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions | |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL3 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC on Port 2. |
| eUICC | No Notification is stored in the eUICC's Pending Notifications List. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC8 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x91YY |
| 2 | S\_Device 🡪eUICC | FETCH 'YY' | LSI COMMAND (“UICC Platform Reset”) |
| 3 | Repeat IC1 to IC3 | | |
| 4 | Repeat IC8 | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DE1 SW = 0x9000  NOTE : A Disable Notification for PROFILE\_OPERATIONAL1 MAY be also present in the response. |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  }  SW=0x9000 |

Test Sequence #09 Nominal: eUICC Memory Reset (with 2 Enabled Profile) while 2 proactive session is ongoing with Terminal Response outstanding – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions | |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 1. |
| eUICC | The PROFILE\_OPERATIONAL3 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 is Enabled on the eUICC on Port 2. |
| eUICC | No Notification is stored in the eUICC's Pending Notifications List. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC1\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| IC8 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91YY |
| IC9 | S\_Device → eUICC | FETCH 'YY' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| IC10 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC11 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x91ZZ |
| 3 | S\_Device 🡪eUICC | FETCH 'ZZ' | LSI COMMAND (“UICC Platform Reset”) |
| 4 | Repeat IC1 to IC3 | | |
| 5 | Repeat IC11 | | |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DE1 SW = 0x9000  NOTE : A Disable Notification for PROFILE\_OPERATIONAL1 and PROFILE\_OPERATIONAL3 MAY be also present in the response. |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  }  SW=0x9000 |

##### 4.2.24.2.4 TC\_eUICC\_ES10c.eUICCMemoryReset\_ErrorCases\_MEPA1

Test Sequence #01 Error: Nothing to delete

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | No Profile is loaded on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | resp EuiccMemoryResetResponse::= {  resetResult nothingToDelete  }  SW=0x9000 |

Test Sequence #02 Error: eUICC Memory Reset while 2 proactive session is ongoing – catBusy supported

|  |  |
| --- | --- |
| Initial Conditions | |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “010203”,  2) | Verify  <MEP\_MODE> = 01,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| IC8 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91YY |
| IC9 | Do not send FETCH command | | |
| IC10 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC11 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | resp EuiccMemoryResetResponse::= {  resetResult catBusy  }SW=0x9000 |
| 2 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| 3 | S\_Device → eUICC | FETCH 'YY' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 4 | S\_Device → eUICC | Send Terminal Response | SW=0x9000 |
| 5 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 6 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 7 | S\_Device → eUICC | Send Terminal Response | SW=0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |

Test Sequence #03 Error: VOID

##### 4.2.24.2.5 TC\_eUICC\_ES10c.eUICCMemoryReset\_MEPA2

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | eUICC is MEP capable |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |
| eUICC | The Default SM-DP+ Address #TEST\_DP\_ADDRESS1 has been set on the ISD-R. |

Test Sequence #01 Nominal: Reset All Operational Profiles (without Enabled Profile)

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |
| eUICC | No Notification is stored in the eUICC's Pending Notifications List. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_EUICC\_INFO2) | Retrieve free non-volatile memory value (tag 0x82) from <EXT\_CARD\_RESOURCE> in EUICCInfo2 as <FREE\_MEM\_OP\_PROF\_INSTALLED> |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk: {  }  SW=0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DE1\_DE2 SW = 0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_EUICC\_INFO2) | Retrieve free non-volatile memory value (tag 0x82) from <EXT\_CARD\_RESOURCE> in EUICCInfo2 as <FREE\_MEMORY\_NO\_PROFILE>  Verify that <FREE\_MEM\_OP\_PROF\_INSTALLED> is lower than <FREE\_MEMORY\_NO\_PROFILE> |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_EUICC\_CONFIGURED\_DATA) | #R\_ES10a\_GECA\_DS\_DP\_1  SW = 0x9000 |

Test Sequence #02 Nominal: Reset the Default SM-DP+ Address only

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |
| eUICC | The Nickname of the PROFILE\_OPERATIONAL3 is equal to #NICKNAME3. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_DEF\_SMDPADDRESS) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk: {  #PROFILE\_INFO1\_DISABLED,  #PROFILE\_INFO3  }  SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_EUICC\_CONFIGURED\_DATA) | #R\_ES10a\_GECA\_DS  SW = 0x9000 |

Test Sequence #03 Nominal: Reset All Operational Profiles and the Default SM-DP+ Address

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk: {  }  SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_EUICC\_CONFIGURED\_DATA) | #R\_ES10a\_GECA\_DS  SW = 0x9000 |

Test Sequence #04 Nominal: Reset All Operational Profiles (with 2 Enabled Profile)

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC. |
| eUICC | No Notification is stored in the eUICC’s Pending Notifications List. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x91XX |
| 2 | S\_Device 🡪eUICC | FETCH ‘XX’ | LSI COMMAND (“UICC Platform Reset”) |
| 3 | Repeat IC1 to IC4 | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DE1\_DE2 SW = 0x9000  NOTE : A Disable Notification for PROFILE\_OPERATIONAL1 and PROFILE\_OPERATIONAL2 MAY be also present in the response. |
| 5 | S\_LPAd 🡪 eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  }  SW=0x9000 |

Test Sequence #05 Nominal: Reset the Default SM-DP+ Address only (with 2 Enabled Profile)

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 is Enabled on the eUICC. |
| eUICC | The Nickname of the PROFILE\_OPERATIONAL3 is equal to #NICKNAME3. |
| eUICC | No Notification is stored in the eUICC's Pending Notifications List. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_DEF\_SMDPADDRESS) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x9000 |
| 2 | S\_LPAd 🡪 eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO3\_ENABLED;  }  SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_EUICC\_CONFIGURED\_DATA) | #R\_ES10a\_GECA\_DS  SW = 0x9000 |

Test Sequence #06 Nominal: eUICC Memory Reset, 1 Operational Profile Enabled, proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions | |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on target port <MEP-A2\_TARGET\_ESIM\_PORT> assigned by the eUICC. |
| eUICC | No Notification is stored in the eUICC's Pending Notifications List. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(<MEP-A2\_TARGET\_ESIM\_PORT>) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC8 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x91YY |
| 2 | S\_Device 🡪eUICC | FETCH 'YY' | LSI COMMAND (“UICC Platform Reset”) |
| 3 | Repeat IC1 to IC3 | | |
| 4 | Repeat IC8 | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DE1 SW = 0x9000  NOTE : A Disable Notification for PROFILE\_OPERATIONAL1 MAY be also present in the response. |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  }  SW=0x9000 |

Test Sequence #07 Nominal: eUICC Memory Reset, 2 Operational Profile Enabled, 2 proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions | |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC. |
| eUICC | No Notification is stored in the eUICC's Pending Notifications List. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| IC8 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91YY |
| IC9 | Do not send FETCH command | | |
| IC10 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC11 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x91ZZ |
| 2 | S\_Device 🡪eUICC | FETCH 'ZZ' | LSI COMMAND (“UICC Platform Reset”) |
| 3 | Repeat IC1 to IC3 | | |
| 4 | Repeat IC11 | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DE1\_DE2 SW = 0x9000  NOTE : A Disable Notification for PROFILE\_OPERATIONAL1 and PROFILE\_OPERATIONAL2 MAY be also present in the response. |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  }  SW=0x9000 |

Test Sequence #08 Nominal: eUICC Memory Reset (with Enabled Profile) while proactive session is ongoing with Terminal Response outstanding – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions | |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on <MEP-A2\_TARGET\_ESIM\_PORT> assigned by the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |
| eUICC | No Notification is stored in the eUICC's Pending Notifications List. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(<MEP-A2\_TARGET\_ESIM\_PORT>) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC8 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x91YY |
| 2 | S\_Device 🡪eUICC | FETCH 'YY' | LSI COMMAND (“UICC Platform Reset”) |
| 3 | Repeat IC1 to IC3 | | |
| 4 | Repeat IC8 | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DE1 SW = 0x9000  NOTE : A Disable Notification for PROFILE\_OPERATIONAL1 MAY be also present in the response. |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  }  SW=0x9000 |

Test Sequence #09 Nominal: eUICC Memory Reset (with 2 Enabled Profile) while 2 proactive session is ongoing with Terminal Response outstanding – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions | |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 is Enabled on the eUICC. |
| eUICC | No Notification is stored in the eUICC's Pending Notifications List. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| IC8 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91YY |
| IC9 | S\_Device → eUICC | FETCH 'YY' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| IC10 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC11 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x91ZZ |
| 2 | S\_Device 🡪eUICC | FETCH 'ZZ' | LSI COMMAND (“UICC Platform Reset”) |
| 3 | Repeat IC1 to IC3 | | |
| 4 | Repeat IC11 | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DE1 SW = 0x9000  NOTE : A Disable Notification for PROFILE\_OPERATIONAL1 and PROFILE\_OPERATIONAL3 MAY be also present in the response. |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  }  SW=0x9000 |

##### 4.2.24.2.6 TC\_eUICC\_ES10c.eUICCMemoryReset\_ErrorCases\_MEPA2

Test Sequence #01 Error: Nothing to delete

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | No Profile is loaded on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | resp EuiccMemoryResetResponse::= {  resetResult nothingToDelete  }  SW=0x9000 |

Test Sequence #02 Error: eUICC Memory Reset while 2 proactive session is ongoing – catBusy supported

|  |  |
| --- | --- |
| Initial Conditions | |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “020103”,  2) | Verify  <MEP\_MODE> = 02,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| IC8 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91YY |
| IC9 | Do not send FETCH command | | |
| IC10 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC11 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | resp EuiccMemoryResetResponse::= {  resetResult catBusy  }SW=0x9000 |
| 2 | PROC\_MEP\_LSI\_MULTIPLEXING(2) | | |
| 3 | S\_Device → eUICC | FETCH 'YY' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 4 | S\_Device → eUICC | Send Terminal Response | SW=0x9000 |
| 5 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 6 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 7 | S\_Device → eUICC | Send Terminal Response | SW=0x9000 |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1;  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |

Test Sequence #03 Error: VOID

##### 4.2.24.2.7 TC\_eUICC\_ES10c.eUICCMemoryReset\_MEPB

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | eUICC is MEP capable |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC. |
| eUICC | The Default SM-DP+ Address #TEST\_DP\_ADDRESS1 has been set on the ISD-R. |

Test Sequence #01 Nominal: Reset All Operational Profiles (without Enabled Profile)

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Disabled on the eUICC. |
| eUICC | No Notification is stored in the eUICC's Pending Notifications List. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030102”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_EUICC\_INFO2) | Retrieve free non-volatile memory value (tag 0x82) from <EXT\_CARD\_RESOURCE> in EUICCInfo2 as <FREE\_MEM\_OP\_PROF\_INSTALLED> |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk: {  }  SW=0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DE1\_DE2 SW = 0x9000 |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_EUICC\_INFO2) | Retrieve free non-volatile memory value (tag 0x82) from <EXT\_CARD\_RESOURCE> in EUICCInfo2 as <FREE\_MEMORY\_NO\_PROFILE>  Verify that <FREE\_MEM\_OP\_PROF\_INSTALLED> is lower than <FREE\_MEMORY\_NO\_PROFILE> |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_EUICC\_CONFIGURED\_DATA) | #R\_ES10a\_GECA\_DS\_DP\_1  SW = 0x9000 |

Test Sequence #02 Nominal: Reset the Default SM-DP+ Address only

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |
| eUICC | The Nickname of the PROFILE\_OPERATIONAL3 is equal to #NICKNAME3. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030102”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_DEF\_SMDPADDRESS) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk: {  #PROFILE\_INFO1\_DISABLED,  #PROFILE\_INFO3  }  SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_EUICC\_CONFIGURED\_DATA) | #R\_ES10a\_GECA\_DS  SW = 0x9000 |

Test Sequence #03 Nominal: Reset All Operational Profiles and the Default SM-DP+ Address

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030102”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk: {  }  SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_EUICC\_CONFIGURED\_DATA) | #R\_ES10a\_GECA\_DS  SW = 0x9000 |

Test Sequence #04 Nominal: Reset All Operational Profiles (with 2 Enabled Profile)

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the Euicc on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 1. |
| eUICC | No Notification is stored in the eUICC’s Pending Notifications List. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030102”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x91XX |
| 2 | S\_Device 🡪eUICC | FETCH ‘XX’ | LSI COMMAND (“UICC Platform Reset”) |
| 3 | Repeat IC1 to IC4 | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DE1\_DE2 SW = 0x9000  NOTE : A Disable Notification for PROFILE\_OPERATIONAL1 and PROFILE\_OPERATIONAL2 MAY be also present in the response. |
| 5 | S\_LPAd 🡪 eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  }  SW=0x9000 |

Test Sequence #05 Nominal: Reset the Default SM-DP+ Address only (with 2 Enabled Profile)

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL3 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 is Enabled on the eUICC on Port 1. |
| eUICC | The Nickname of the PROFILE\_OPERATIONAL3 is equal to #NICKNAME3. |
| eUICC | No Notification is stored in the eUICC's Pending Notifications List. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030102”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_DEF\_SMDPADDRESS) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x9000 |
| 2 | S\_LPAd 🡪 eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1,  #PROFILE\_INFO3\_MEPB;  }  SW=0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #GET\_EUICC\_CONFIGURED\_DATA) | #R\_ES10a\_GECA\_DS  SW = 0x9000 |

Test Sequence #06 Nominal: eUICC Memory Reset, 1 Operational Profile Enabled, proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions | |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | No Notification is stored in the eUICC's Pending Notifications List. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030102”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| NOTE: for clarity, the current Port is already Port 0 | | | |
| IC4 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC5 | Do not send FETCH command | | |
| IC6 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x91YY |
| 2 | S\_Device 🡪eUICC | FETCH 'YY' | LSI COMMAND (“UICC Platform Reset”) |
| 3 | Repeat IC1 to IC3 | | |
| 4 | Repeat IC6 | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DE1 SW = 0x9000  NOTE : A Disable Notification for PROFILE\_OPERATIONAL1 MAY be also present in the response. |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  }  SW=0x9000 |

Test Sequence #07 Nominal: eUICC Memory Reset, 2 Operational Profile Enabled, 2 proactive session is ongoing – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions | |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 1. |
| eUICC | No Notification is stored in the eUICC's Pending Notifications List. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030102”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC8 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91YY |
| IC9 | Do not send FETCH command | | |
| IC10 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC11 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x91ZZ |
| 2 | S\_Device 🡪eUICC | FETCH 'ZZ' | LSI COMMAND (“UICC Platform Reset”) |
| 3 | Repeat IC1 to IC3 | | |
| 4 | Repeat IC11 | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DE1\_DE2 SW = 0x9000  NOTE : A Disable Notification for PROFILE\_OPERATIONAL1 and PROFILE\_OPERATIONAL2 MAY be also present in the response. |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse ::= profileInfoListOk : {  }  SW=0x9000 |

Test Sequence #08 Nominal: eUICC Memory Reset (with Enabled Profile) while proactive session is ongoing with Terminal Response outstanding – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions | |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL3 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 is Disabled on the eUICC. |
| eUICC | No Notification is stored in the eUICC's Pending Notifications List. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030102”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| IC7 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x9000 |
| 2 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x91YY |
| 3 | S\_Device 🡪eUICC | FETCH 'YY' | LSI COMMAND (“UICC Platform Reset”) |
| 4 | Repeat IC1 to IC4 | | |
| 5 | Repeat IC7 | | |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DE1 SW = 0x9000  NOTE : A Disable Notification for PROFILE\_OPERATIONAL1 MAY be also present in the response. |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse ::= profileInfoListOk : {  }  SW=0x9000 |

Test Sequence #09 Nominal: eUICC Memory Reset (with 2 Enabled Profile) while 2 proactive session is ongoing with Terminal Response outstanding – catBusy not supported

|  |  |
| --- | --- |
| Initial Conditions | |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL3 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL3 is Enabled on the eUICC on Port 1. |
| eUICC | No Notification is stored in the eUICC's Pending Notifications List. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030102”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC8 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91YY |
| IC9 | S\_Device → eUICC | FETCH 'YY' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| IC10 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | #R\_EUICC\_MEMORY\_RESET\_OK  SW=0x9000 |
| 2 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x91ZZ |
| 3 | S\_Device 🡪eUICC | FETCH 'ZZ' | LSI COMMAND (“UICC Platform Reset”) |
| 4 | Repeat IC1 to IC4 | | |
| 5 | Repeat IC10 | | |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DE1 SW = 0x9000  NOTE : A Disable Notification for PROFILE\_OPERATIONAL1 and PROFILE\_OPERATIONAL3 MAY be also present in the response. |
| 7 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  }  SW=0x9000 |

##### 4.2.24.2.8 TC\_eUICC\_ES10c.eUICCMemoryReset\_ErrorCases\_MEPB

Test Sequence #01 Error: Nothing to delete

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | No Profile is loaded on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030102”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | resp EuiccMemoryResetResponse::= {  resetResult nothingToDelete  }  SW=0x9000 |

Test Sequence #02 Error: eUICC Memory Reset while 2 proactive session is ongoing – catBusy supported

|  |  |
| --- | --- |
| Initial Conditions | |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled on the eUICC on Port 0. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 is Enabled on the eUICC on Port 1. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | RESET | Extract <ATR>  Verify ‘LSI Support’ is present in <ATR> |
| IC2 | S\_Device | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP (  2,  #IUT\_MEP\_LSI\_OPTIONS,  “030102”,  2) | Verify  <MEP\_MODE> = 03,  Verify  <MEP\_LSI\_OPTION> = #IUT\_MEP\_LSI\_OPTIONS,  Verify  <MEP\_MAX\_LSIS> <= #IUT\_MEP\_MAX\_LSIS |
| IC3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | | |
| IC4 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| IC5 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC6 | Do not send FETCH command | | |
| IC7 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| IC8 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91YY |
| IC9 | Do not send FETCH command | | |
| IC10 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #EUICC\_MEMORY\_RESET\_OP\_PRO) | resp EuiccMemoryResetResponse::= {  resetResult catBusy  }SW=0x91YY |
| 2 | S\_Device → eUICC | FETCH 'YY' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 3 | S\_Device → eUICC | Send Terminal Response | SW=0x9000 |
| 4 | PROC\_MEP\_LSI\_MULTIPLEXING(0) | | |
| 5 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 6 | S\_Device → eUICC | Send Terminal Response | SW=0x9000 |
| 7 | PROC\_MEP\_LSI\_MULTIPLEXING(1) | | |
| 8 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  <NO\_PARAM>,  <NO\_PARAM>)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_MEPB,  #PROFILE\_INFO2\_ENABLED;  }  SW=0x9000 |

Test Sequence #03 Error: VOID

### 4.2.25 ES10c (LPA -- eUICC): GetEID

#### 4.2.25.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 3.3.1
* Section 5.7.20

#### 4.2.25.2 Test Cases

##### 4.2.25.2.1 TC\_eUICC\_ES10c.GetEID

Test Sequence #01 Nominal

The purpose of this test is to ensure that it is possible to retrieve the EID.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_EID) | resp GetEuiccDataResponse ::= {  eidValue #EID1  }  SW=0x9000 |

Test Sequence #02 Error

The purpose of this test is to ensure that if the provided tagList is invalid or unsupported, the eUICC returns an error status word.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_EID\_INVALID) | No response data return and SW different than 0x9000 |

### 4.2.26 ES10c (LPA -- eUICC): SetNickname

#### 4.2.26.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 5.7.21

#### 4.2.26.2 Test Cases

##### 4.2.26.2.1 TC\_eUICC\_ES10c.SetNickname

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is loaded on the eUICC. |

Test Sequence #01 Nominal: Add a Nickname to a Disabled Operational Profile

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled. |
| eUICC | The Nickname of the PROFILE\_OPERATIONAL1 is empty. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #SET\_NICKNAME\_OP\_PROF1) | resp SetNicknameResponse ::= {  setNicknameResult ok  }  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   …  profileNickname #NICKNAME2  …  }  }  SW=0x9000 |

Test Sequence #02 Nominal: Update a Nickname of a Disabled Operational Profile

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled. |
| eUICC | The Nickname of the PROFILE\_OPERATIONAL1 is equal to #NICKNAME1. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #SET\_NICKNAME\_OP\_PROF1) | resp SetNicknameResponse ::= {  setNicknameResult ok  }  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   …  profileNickname #NICKNAME2  …  }  }  SW=0x9000 |

Test Sequence #03 Nominal: Remove a Nickname from a Disabled Operational Profile

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled. |
| eUICC | The Nickname of the PROFILE\_OPERATIONAL1 is equal to #NICKNAME1. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #SET\_NICKNAME\_EMPTY\_OP\_PROF1) | resp SetNicknameResponse ::= {  setNicknameResult ok  }  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   …  *-- profileNickname SHALL not*  *-- be present*  …  }  }  SW=0x9000 |

Test Sequence #04 Nominal: Remove a non-existing Nickname from a Disabled Operational Profile

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled. |
| eUICC | The Nickname of the PROFILE\_OPERATIONAL1 is empty. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #SET\_NICKNAME\_EMPTY\_OP\_PROF1) | resp SetNicknameResponse ::= {  setNicknameResult ok  }  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   …  *-- profileNickname SHALL not*  *-- be present*  …  }  }  SW=0x9000 |

Test Sequence #05 Nominal: Add a Nickname to an Enabled Operational Profile

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |
| eUICC | The Nickname of the PROFILE\_OPERATIONAL1 is empty. |

This test sequence SHALL be the same as the Test Sequence #01 defined in this section.

Test Sequence #06 Nominal: Update a Nickname of an Enabled Operational Profile

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |
| eUICC | The Nickname of the PROFILE\_OPERATIONAL1 is equal to #NICKNAME1. |

This test sequence SHALL be the same as the Test Sequence #02 defined in this section.

Test Sequence #07 Nominal: Remove a Nickname from an Enabled Operational Profile

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |
| eUICC | The Nickname of the PROFILE\_OPERATIONAL1 is equal to #NICKNAME1. |

This test sequence SHALL be the same as the Test Sequence #03 defined in this section.

Test Sequence #08 Nominal: Remove a non-existing Nickname from an Enabled Operational Profile

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |
| eUICC | The Nickname of the PROFILE\_OPERATIONAL1 is empty. |

This test sequence SHALL be the same as the Test Sequence #04 defined in this section.

Test Sequence #09 Error: ICCID not found

The purpose of this test is to ensure that the method ES10c.SetNickname returns an error in case the targeted Profile does not exist on the eUICC.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled. |
| eUICC | The Profile identified by the ICCID #ICCID\_UNKNOWN is not present on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #SET\_NICKNAME\_ICCID\_UNKNOWN) | resp SetNicknameResponse ::= {  setNicknameResult iccidNotFound  }  SW=0x9000 |

### 4.2.27 ES10b (LPA -- eUICC): GetRAT

#### 4.2.27.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 2.4a.1.2
* Section 2.9.2.1, 2.9.2.3, 2.9.3.3
* Section 5.7.22

#### 4.2.27.2 Test Cases

##### 4.2.27.2.1 TC\_eUICC\_ES10b.GetRAT

Test Sequence #01 Nominal: Get Default RAT

The purpose of this test is to verify that the eUICC can be configured with a RAT as defined in SGP.22 [2] section 2.9.2.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The EUM has configured the eUICC's RAT as defined in section G.2.4. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_RAT) | #R\_DEFAULT\_RAT  SW = 0x9000 |

Test Sequence #02 Nominal: With additional PPARs

The purpose of this test is to verify that the eUICC can be configured with a RAT that contains custom rules reflecting agreements between some Operators and OEMs. After having checked the content of the RAT, Profiles with PPR1 and PPR2 are installed in order to make sure that the eUICC accepts such PPRs.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The EUM has configured the eUICC's RAT as defined in section G.2.5. |
| eUICC | There is no Profile installed in the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_RAT) | #R\_RAT\_WITH\_OTHER\_RULES with exact same structure and order  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | Install PROFILE\_OPERATIONAL4 | Profile successfully downloaded (i.e. ProfileInstallationResult contains a SuccessResult) |
| 3 | S\_LPAd → eUICC | Delete PROFILE\_OPERATIONAL4 |  |
| 4 | S\_LPAd → eUICC | Install PROFILE\_OPERATIONAL3 | Profile successfully downloaded (i.e. ProfileInstallationResult contains a SuccessResult) |

### 4.2.28 ES10b (LPA -- eUICC): LoadRPMPackage

#### 4.2.28.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 2.10, 2.10.1, 2.10.2
* Section 5.1
* Section 5.7.14a

#### 4.2.28.2 Test Cases

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is loaded on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled. |
| eUICC | The communication between the S\_Device and the eUICC has been initialized and the S\_LPAd has selected the ISD-R.  Common Mutual Authentication procedure has been successfully executed between the eUICC and the S\_SM-DP+ with the CtxParams1 as #CTX\_PARAMS1\_RPM\_ICCID1 |

##### 4.2.28.2.1 TC\_eUICC\_ES10b.LoadRPMPackage\_EnableProfile

Test Sequence #01 Nominal: RPM Command EnableProfile - no profiles enabled already

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_EN. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  enable,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x91XX with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  enableResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  0, -- OK response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_Device → eUICC | FETCH 'XX' | REFRESH Command  (“UICC Reset”) |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 6 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 7 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

Test Sequence #02 Error: RPM Command EnableProfile – ICCID unknown

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been loaded with #METADATA\_OP\_PROF1\_RPM\_CONF\_EN. |
| eUICC | The Operational Profile identified by the ICCID #ICCID\_OP\_PROFX is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  enable,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROFX,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  enableResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROFX,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  commandError  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #03 Error: RPM Command EnableProfile – SM-DP+ OID not in Managing SM-DP+ List

The purpose of this Test Secuence is to ensure RPM Command EnableProfile is not executed if the SM-DP+ that sent the RPM Command is not included in the Managing SM-DP+ List in the Profile Metadata (rpmConfiguration).

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been loaded on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_EN\_DP\_OID2. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT (  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  enable,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  enableResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  commandError  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (  MTD\_GET\_PROFILE\_INFO ( #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #04 Error: RPM Command EnableProfile – Allowed CI Public Key Identifier not matched

The purpose of this test sequence is to ensure RPM Command EnableProfile is not executed if Subject Key Identifier of the CI corresponding to CERT.DPauth.SIG attached to the ongoing session does not match with the Allowed CI Public Key Identifier in the Profile Metadata (rpmConfiguration).

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_OP\_PROF1\_RPM\_CONF\_EN\_CI\_PKI\_RAND is loaded on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT (  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  enable,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  enableResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  commandError  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (  MTD\_GET\_PROFILE\_INFO (  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #05 Error: RPM Command EnableProfile – Another Profile currently Enabled

The purpose of this test sequence is to ensure RPM Command EnableProfile is not executed while another Profile is Enabled.

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been loaded with #METADATA\_OP\_PROF1\_RPM\_CONF\_EN. |
| eUICC | The PROFILE\_OPERATIONAL2 with #METADATA\_OP\_PROF2\_RPM\_CONF\_EN\_OWNER\_OID1 (without PPR1 present) is loaded and Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  enable,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with unsuccessful enableResult in the response data  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  #PROFILE\_INFO2\_ENABLED  }  SW=0x9000 |

Test Sequence #06 Error: RPM Command EnableProfile – Profile is not in Disabled state

The purpose of this Test Sequence is to ensure RPM Command EnableProfile is not executed if the target Profile is not in Disabled state.

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been loaded on the eUICC with  #METADATA\_OP\_PROF1\_RPM\_CONF\_EN and it is Enabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT (  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  enable,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  enableResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  profileNotInDisabledState  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (  MTD\_GET\_PROFILE\_INFO (  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1}  SW=0x9000 |

Test Sequence #07 Error: RPM Command EnableProfile – disallowed by policy

The purpose of this Test Sequence is to ensure RPM Command EnableProfile is not executed if the currently Enabled Profile cannot be disabled due to Profile Policy Rule.

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been loaded on the eUICC with  #METADATA\_OP\_PROF1\_RPM\_CONF\_EN. |
| eUICC | The PROFILE\_OPERATIONAL2 with #METADATA\_OP\_PROF2\_RPM\_CONF\_EN\_OWNER\_OID1\_PPR1 has been loaded before the PROFILE\_OPERATIONAL1 and is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT (  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  enable,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  enableResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  disallowedByPolicy  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (  MTD\_GET\_PROFILE\_INFO (  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #08 Error: RPM Command EnableProfile – Invalid Transaction Id

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been loaded with #METADATA\_OP\_PROF1\_RPM\_CONF\_EN. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  enable,  <INVALID\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  )  -- The <INVALID\_TRANSACTION\_ID> SHALL NOT be equal to the Transaction Id in the AuthenticateServerRequest | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  enableResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  3, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  invalidTransactionId  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG  • The <S\_TRANSACTION\_ID> SHALL be equal to the one set in the AuthenticateServerRequest. |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #09 Error: RPM Command EnableProfile – Invalid Signature

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been loaded with #METADATA\_OP\_PROF1\_RPM\_CONF\_EN. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  enable,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  enableResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  3, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  invalidSignature  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #10 Error: RPM Command EnableProfile – No RPM Session on going

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been loaded with #METADATA\_OP\_PROF1\_RPM\_CONF\_EN |
| eUICC | No RPM session is on-going (i.e. No common mutual authentication procedure has been completed for a RPM session ) |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  enable,  <INVALID\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  )  *-- The <INVALID\_TRANSACTION\_ID> any Transaction Id.* | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  NO\_PARAM,  <S\_TRANSACTION\_ID>,  NO\_PARAM,  4, -- error response, LoadRpmPackageErrorCodeNotSigned  NO\_PARAM,  NO\_PARAM,  NO\_PARAM,  NO\_PARAM,  noSession  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

##### 4.2.28.2.2 TC\_eUICC\_ES10b.LoadRPMPackage\_DisableProfile

Test Sequence #01 Nominal: RPM Command DisableProfile – by ICCID

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_DI.  (PPR1 is not set in the Metadata) |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  disable,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x91XX with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  disableResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  0, -- OK response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_Device → eUICC | FETCH 'XX' | REFRESH Command  (“UICC Reset”) |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 6 | S\_Device → eUICC | [SELECT\_ICCID] | SW=6A82 |

Test Sequence #02 Error: RPM Command DisableProfile – ICCID unknown

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been loaded with #METADATA\_OP\_PROF1\_RPM\_CONF\_DI.  (PPR1 is not set in the Metadata) |
| eUICC | The PROFILE\_OPERATIONAL1 has been Enabled. |
| eUICC | The Operational Profile identified by the ICCID #ICCID\_OP\_PROFX is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  disable,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROFX,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  disableResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROFX,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  commandError  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #03 Error: RPM Command DisableProfile – SM-DP+ OID not in Managing SM-DP+ List

The purpose of this Test Sequence is to ensure RPM Command DisableProfile is not executed if the SM-DP+ that sent the RPM Command is not included in the Managing SM-DP+ List in the Profile Metadata (rpmConfiguration).

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been loaded with #METADATA\_OP\_PROF1\_RPM\_CONF\_DI\_DP\_OID2 and is Enabled on the eUICC (PPR1 is not set in the Metadata).  NOTE: #S\_SM\_DP+\_OID2 set in the Profile Metadata is different from SM-DP+ OID in CERT.DPauth.SIG attached to the ongoing RSP session. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT (  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  disable,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  disableResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  commandError  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (  MTD\_GET\_PROFILE\_INFO (  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #04 Error: RPM Command DisableProfile – Allowed CI Public Key Identifier not matched

The purpose of this test sequence is to ensure RPM Command DisableProfile is not executed if Subject Key Identifier of the CI corresponding to CERT.DPauth.SIG attached to the ongoing session does not match with the Allowed CI Public Key Identifier in the Profile Metadata (rpmConfiguration).

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_OP\_PROF1\_RPM\_CONF\_DI\_CI\_PKI\_RAND is loaded and Enabled on the eUICC.  (PPR1 is not set in the Metadata) |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT (  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  disable,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  disableResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  commandError  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (  MTD\_GET\_PROFILE\_INFO (  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #05 Error: RPM Command DisableProfile – Profile is not in Enabled state

The purpose of this Test Sequence is to ensure RPM Command DisableProfile is not executed if the target Profile is not in Enabled state.

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been loaded on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_DI and it is in Disabled state. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT (  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  disable,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  disableResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  profileNotInEnabledState  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (  MTD\_GET\_PROFILE\_INFO (  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED}  SW=0x9000 |

Test Sequence #06 Error: RPM Command DisableProfile – disallowed by policy

The purpose of this Test Sequence is to ensure RPM Command DisableProfile is not executed if the currently Enabled Profile cannot be disabled due to Profile Policy Rule.

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been loaded with #METADATA\_OP\_PROF1\_RPM\_CONF\_DI\_PPR1 and is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT (  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  disable,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  disableResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  disallowedByPolicy  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (  MTD\_GET\_PROFILE\_INFO (  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

##### 4.2.28.2.3 TC\_eUICC\_ES10b.LoadRPMPackage\_DeleteProfile

Test Sequence #01 Nominal: RPM Command DeleteProfile – by ICCID

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_DE.  (PPR2 is not set in the Metadata) |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  delete,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data  MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  deleteResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  0, -- OK response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse ::= profileInfoListOk : {  }  SW=0x9000 |

Test Sequence #02 Error: RPM Command DeleteProfile\_ICCID unknown

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been loaded with #METADATA\_OP\_PROF1\_RPM\_CONF\_DE.  (PPR2 is not set in the Metadata) |
| eUICC | The Operational Profile identified by the ICCID #ICCID\_OP\_PROFX is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  delete,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROFX,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  deleteResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROFX,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  commandError  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #03 Error: RPM Command DeleteProfile\_SM-DP+ OID not in Managing SM-DP+ List

The purpose of this Test Sequence is to ensure RPM Command DeleteProfile is not executed if the SM-DP+ that sends the RPM Command is not included in the Managing SM-DP+ List in the Profile Metadata (rpmConfiguration).

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been loaded with #METADATA\_OP\_PROF1\_RPM\_CONF\_DE\_DP\_OID2.  (PPR2 is not set in the Metadata)  NOTE: #S\_SM\_DP+\_OID2 set in the Profile Metadata is different from SM-DP+ OID in CERT.DPauth.SIG attached to the ongoing RSP session. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT (  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  delete,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  deleteResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  commandError  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (  MTD\_GET\_PROFILE\_INFO (  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #04 Error: RPM Command DeleteProfile – Allowed CI Public Key Identifier not matched

The purpose of this test sequence is to ensure RPM Command DeleteProfile is not executed if Subject Key Identifier of the CI corresponding to CERT.DPauth.SIG attached to the ongoing session does not match with the Allowed CI Public Key Identifier in the Profile Metadata (rpmConfiguration).

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_OP\_PROF1\_RPM\_CONF\_DE\_CI\_PKI\_RAND is loaded on the eUICC.  (PPR2 is not set in the Metadata) |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT (  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  delete,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  deleteResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  commandError  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (  MTD\_GET\_PROFILE\_INFO (  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #05 Error: RPM Command DeleteProfile – Profile is not in Disabled state

The purpose of this Test Sequence is to ensure RPM Command DeleteProfile is not executed if the target Profile is in Enabled state.

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been loaded on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_DE and it is in Enabled state.  (PPR2 is not set in the Metadata) |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT (  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  delete,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  deleteResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  profileNotInDisabledState  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (  MTD\_GET\_PROFILE\_INFO (  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1}  SW=0x9000 |

Test Sequence #06 Error: RPM Command DeleteProfile – disallowed by policy

The purpose of this Test Sequence is to ensure RPM Command DeleteProfile is not executed if the currently Disabled Profile can not be Deleted due to Profile Prolicy Rule.

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been loaded with #METADATA\_OP\_PROF1\_RPM\_CONF\_DE\_PPR2 and is Disabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT (  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  delete,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  deleteResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  disallowedByPolicy  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA (  MTD\_GET\_PROFILE\_INFO (  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

##### 4.2.28.2.4 TC\_eUICC\_ES10b.LoadRPMPackage\_ListProfileInfo

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_ALL\_PPR1. |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed on the eUICC with #METADATA\_OP\_PROF2\_RPM\_CONF\_ALL\_OWNER2. |
| eUICC | The PROFILE\_OPERATIONAL3 has been installed on the eUICC with #METADATA\_OP\_PROF3\_RPM\_CONF\_ALL. |
| eUICC | The Nickname of the PROFILE\_OPERATIONAL3 is #NICKNAME3. |
| eUICC | The PROFILE\_OPERATIONAL1 is enabled. |

Test Sequence #01 Nominal: RPM Command\_ListProfileInfo\_byICCID

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_FOR\_LIST\_PROFILE\_INFO (  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  listProfileInfoResult,  <S\_TRANSACTION\_ID>,  NO\_PARAM,  0, -- OK response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  #PROFILE\_INFO1,  NO\_PARAM,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |

Test Sequence #02 Nominal: RPM Command\_ListProfileInfo\_byProfileOwnerOID

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_FOR\_LIST\_PROFILE\_INFO (  <S\_TRANSACTION\_ID>,  NO\_PARAM,  <S\_SM\_DP+\_SIGNATURE3>,  #S\_PROFILE\_OWNER\_OID,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  listProfileInfoResult,  <S\_TRANSACTION\_ID>,  NO\_PARAM,  0, -- OK response  {  #NOTIF\_METADATA\_PROF1\_DP1\_RPR  },  #S\_SM\_DP+\_OID,  { #PROFILE\_INFO1,  #PROFILE\_INFO3 },  NO\_PARAM,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |

Test Sequence #03 Nominal: RPM Command ListProfileInfo - ICCID and tagList PPR present

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_FOR\_LIST\_PROFILE\_INFO (  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  ‘99’H  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  listProfileInfoResult,  <S\_TRANSACTION\_ID>,  NO\_PARAM,  0, -- OK response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  { PROFILES\_INFO\_ICCID\_TAGLIST6},  NO\_PARAM,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |

Test Sequence #04 Nominal: RPM Command\_ListProfileInfo\_- Profile Owner ID and tagList LPR Configuration present

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_FOR\_LIST\_PROFILE\_INFO (  <S\_TRANSACTION\_ID>,  NO\_PARAMS,  <S\_SM\_DP+\_SIGNATURE3>,  #S\_PROFILE\_OWNER\_OID,  ‘BA5A’H  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  listProfileInfoResult,  <S\_TRANSACTION\_ID>,  NO\_PARAM,  0, -- OK response  {  #NOTIF\_METADATA\_PROF1\_DP1\_RPR  },  #S\_SM\_DP+\_OID,  { PROFILES\_INFO\_RPM\_TAGLIST1,  PROFILES\_INFO\_RPM\_TAGLIST2 },  NO\_PARAM,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |

Test Sequence #05 Nominal: RPM Command\_ListProfileInfo\_- Profile Owner ID and tagList with multiple Tags present

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_FOR\_LIST\_PROFILE\_INFO (  <S\_TRANSACTION\_ID>,  NO\_PARAM,  <S\_SM\_DP+\_SIGNATURE3>,  #S\_PROFILE\_OWNER\_OID,  ‘BA9BBC5A’H  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  listProfileInfoResult,  <S\_TRANSACTION\_ID>,  NO\_PARAM,  0, -- OK response  {  #NOTIF\_METADATA\_PROF1\_DP1\_RPR  },  #S\_SM\_DP+\_OID,  { PROFILES\_INFO\_RPM\_TAGLIST3,  PROFILES\_INFO\_RPM\_TAGLIST4 },  NO\_PARAM,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |

Test Sequence #06 Nominal: RPM Command\_ListProfileInfo\_- ICCID specified and tagList with multiple Tags present

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_FOR\_LIST\_PROFILE\_INFO (  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF2,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  ‘5ABA9F1F’H  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  listProfileInfoResult,  <S\_TRANSACTION\_ID>,  NO\_PARAM,  0, -- OK response  {  #NOTIF\_METADATA\_PROF1\_DP1\_RPR  },  #S\_SM\_DP+\_OID,  { PROFILES\_INFO\_RPM\_TAGLIST5 },  NO\_PARAM,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |

Test Sequence #07 Nominal: RPM Command\_ListProfileInfo\_- ICCID specified, no profile found

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_FOR\_LIST\_PROFILE\_INFO (  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROFX,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  ‘5A’H  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  listProfileInfoResult,  <S\_TRANSACTION\_ID>,  NO\_PARAM,  0, -- OK response  {  #NOTIF\_METADATA\_PROF1\_DP1\_RPR  },  #S\_SM\_DP+\_OID,  {},  NO\_PARAM,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |

Test Sequence #08 Nominal: RPM Command\_ListProfileInfo\_- Both ICCID and TagList specified, TagList not present

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_FOR\_LIST\_PROFILE\_INFO (  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  ‘BD9F1F’H  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  listProfileInfoResult,  <S\_TRANSACTION\_ID>,  NO\_PARAM,  0, -- OK response  {  #NOTIF\_METADATA\_PROF1\_DP1\_RPR  },  #S\_SM\_DP+\_OID,  {},  NO\_PARAM,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |

Test Sequence #09 Nominal: RPM Command\_ListProfileInfo\_- Profile Owner ID specified, no profile found

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_FOR\_LIST\_PROFILE\_INFO (  <S\_TRANSACTION\_ID>,  NO\_PARAM,  <S\_SM\_DP+\_SIGNATURE3>,  #S\_PROFILE\_OWNER\_OIDX,  ‘5A’H  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  listProfileInfoResult,  <S\_TRANSACTION\_ID>,  NO\_PARAM,  0, -- OK response  {  #NOTIF\_METADATA\_PROF1\_DP1\_RPR  },  #S\_SM\_DP+\_OID,  {},  NO\_PARAM,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |

Test Sequence #10 Error: RPM Command\_ListProfileInfo\_- ICCID specified, SM-DP+ OID not in Managing SM-DP+ list

The purpose of this Test Secuence is to ensure RPM Command ListProfileInfo returns an empty ProfileInfoListOk if the SM-DP+ that sent the RPM Command is not included in the Managing SM-DP+ List in the Profile Metadata (rpmConfiguration).

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been loaded on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_EN\_DP\_OID2. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_FOR\_LIST\_PROFILE\_INFO (  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  ‘5A’H  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  listProfileInfoResult,  <S\_TRANSACTION\_ID>,  NO\_PARAM,  0, -- OK response  {  #NOTIF\_METADATA\_PROF1\_DP1\_RPR  },  #S\_SM\_DP+\_OID,  {},  NO\_PARAM,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |

Test Sequence #11 Error: RPM Command\_ListProfileInfo\_- ICCID specified, allowed CI public key identifier does not match

The purpose of this test sequence is to ensure RPM Command ListProfileInfo returns an empty ProfileInfoListOk if Subject Key Identifier of the CI corresponding to CERT.DPauth.SIG attached to the ongoing session does not match with the Allowed CI Public Key Identifier in the Profile Metadata (rpmConfiguration).

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_OP\_PROF1\_RPM\_CONF\_ALL\_CI\_PKI\_RAND is loaded on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_FOR\_LIST\_PROFILE\_INFO (  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  ‘5A’H  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  listProfileInfoResult,  <S\_TRANSACTION\_ID>,  NO\_PARAM,  0, -- OK response  {  #NOTIF\_METADATA\_PROF1\_DP1\_RPR  },  #S\_SM\_DP+\_OID,  {},  NO\_PARAM,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |

Test Sequence #12 Error: RPM Command\_ListProfileInfo: ICCID specified, Enable on going

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 corresponds to <ISD\_P\_AID1>. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_Case3(  MTD\_ENABLE\_PROFILE(  NO\_PARAM,  <ISD\_P\_AID1>,  TRUE)) | No response data is returned  SW=0x91XX |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_FOR\_LIST\_PROFILE\_INFO (  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  ‘5A’H  )  ) | SW=0x91xx without response data for all STORE DATA commands except for the last one  SW=0x91xx with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  listProfileInfoResult,  <S\_TRANSACTION\_ID>,  NO\_PARAM,  1, -- error response  {  #NOTIF\_METADATA\_PROF1\_DP1\_RPR  },  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  profileChangeOngoing  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 3 | S\_Device → eUICC | FETCH 'XX' | REFRESH Command (“UICC Reset”) |
| 4 | Repeat IC1 and IC2 | | |

Test Sequence #13 Error: RPM Command\_ListProfileInfo\_- Profile Owner ID specified, SM-DP+ OID not in Managing SM-DP+ list

The purpose of this Test Sequence is to ensure RPM Command ListProfileInfo returns an empty ProfileInfoListOk if the SM-DP+ that sent the RPM Command is not included in the Managing SM-DP+ List in the Profile Metadata (rpmConfiguration).

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been loaded on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_ALL \_DP\_OID2. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_FOR\_LIST\_PROFILE\_INFO (  <S\_TRANSACTION\_ID>,  NO\_PARAM,  <S\_SM\_DP+\_SIGNATURE3>,  #S\_PROFILE\_OWNER\_OID,  ‘5A’H  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  listProfileInfoResult,  <S\_TRANSACTION\_ID>,  NO\_PARAM,  0, -- OK response  {  #NOTIF\_METADATA\_PROF1\_DP1\_RPR  },  #S\_SM\_DP+\_OID,  {},  NO\_PARAM,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| NOTE: #S\_SM\_DP+\_OID2 set in the Profile Metadata is different from SM-DP+ OID in CERT.DPauth.SIG attached to the ongoing RSP session. | | | |

Test Sequence #14 Error: RPM Command\_ListProfileInfo\_- Profile Owner ID specified, allowed CI public key identifier does not match

The purpose of this test sequence is to ensure RPM Command ListProfileInfo returns an empty ProfileInfoListOk if Subject Key Identifier of the CI corresponding to CERT.DPauth.SIG attached to the ongoing session does not match with the Allowed CI Public Key Identifier in the Profile Metadata (rpmConfiguration).

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_OP\_PROF1\_RPM\_CONF\_ALL\_CI\_PKI\_RAND is loaded on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_FOR\_LIST\_PROFILE\_INFO (  <S\_TRANSACTION\_ID>,  NO\_PARAM,  <S\_SM\_DP+\_SIGNATURE3>,  #S\_PROFILE\_OWNER\_OID,  ‘5A’H  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  listProfileInfoResult,  <S\_TRANSACTION\_ID>,  NO\_PARAM,  0, -- OK response  {  #NOTIF\_METADATA\_PROF1\_DP1\_RPR  },  #S\_SM\_DP+\_OID,  {},  NO\_PARAM,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |

##### 4.2.28.2.5 TC\_eUICC\_ES10b.LoadRPMPackage\_UpdateMetadata

Test Sequence #01 Nominal: RPM Command UpdateMetadata – Remove PPR1

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_UPDATE\_MD\_PPR. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  updateMetadata,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  profilePolicyRules {ppr2},  NO\_PARAM  )) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  updateMetadataResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  0, -- OK response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PPR\_OP\_PROF1) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   iccid #ICCID\_OP\_PROF1,  profilePolicyRules {ppr2}  }  }  SW=0x9000 |

Test Sequence #02 Nominal: RPM Command UpdateMetadata – RPM Configuration

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_ALL\_PPR1. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  updateMetadata,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  rpmConfiguration #RPM\_CONFIG\_OP\_PROF1,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  updateMetadataResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  0, -- OK response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_RPM\_CONFIG\_OP\_PROF1) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   iccid #ICCID\_OP\_PROF1,  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable, disable, delete, listProfileInfo}  tagList ‘99BA9BBC’H  }  },  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  }  }  SW=0x9000 |

Test Sequence #03 Nominal: RPM Command UpdateMetadata – HRI Server Address

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_ALL\_PPR1. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  updateMetadata,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  hriServerAddress #TEST\_HRI\_ADDRESS3,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  updateMetadataResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  0, -- OK response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_HRI\_SRV\_ADDRESS\_OP\_PROF1) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   iccid #ICCID\_OP\_PROF1,  hriServerAddress {#TEST\_HRI\_ADDRESS3}  }  }  SW=0x9000 |

Test Sequence #04 Nominal: RPM Command UpdateMetadata – LPR Configuration

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_ALL\_PPR1. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  updateMetadata,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  lprConfiguration { pcmpAddress #TEST\_PCMP\_ADDRESS1 },  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  updateMetadataResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  0, -- OK response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_LPR\_CONFIG\_OP\_PROF1) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   iccid #ICCID\_OP\_PROF1,  lprConfiguration {  pcmpAddress  #TEST\_PCMP\_ADDRESS1  }  }  }  SW=0x9000 |

Test Sequence #05 Nominal: RPM Command UpdateMetadata – Multiple Tags

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_ALL\_PPR1. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  updateMetadata,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  #MULTIPLE\_TAGS\_OP\_PROF1,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  updateMetadataResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  0, -- OK response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_MULTIPLE\_TAGS \_OP\_PROF1) | resp ProfileInfoListResponse ::=  profileInfoListOk : {  { #MULTIPLE\_TAGS\_OP\_PROF1 }  }  SW=0x9000 |

Test Sequence #06 Error: RPM Command UpdateMetadata – ICCID unknown

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_ALL\_PPR1. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  updateMetadata,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROFX,  <S\_SM\_DP+\_SIGNATURE3>,  lprConfiguration { pcmpAddress #TEST\_PCMP\_ADDRESS1 },  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  updateMetadataResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROFX,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  commandError  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_LPR\_CONFIG\_OP\_PROF1) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   iccid #ICCID\_OP\_PROF1  }  }  SW=0x9000 |

Test Sequence #07 Error: RPM Command\_UpdateMetadata – SM-DP+ OID not in Managing SM-DP+ list

The purpose of this Test Sequence is to ensure RPM Command UpdateMetadata is not executed if the SM-DP+ that sent the RPM Command is not included in the Managing SM-DP+ List in the Profile Metadata (rpmConfiguration).

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been loaded on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_ALL\_DP\_OID2. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  updateMetadata,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  lprConfiguration { pcmpAddress #TEST\_PCMP\_ADDRESS1 },  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  updateMetadataResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  commandError  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_LPR\_CONFIG\_OP\_PROF1) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   iccid #ICCID\_OP\_PROF1  }  }  SW=0x9000 |

Test Sequence #08 Error: RPM Command\_UpdateMetadata – TagList not in Managing SM-DP+ list

The purpose of this Test Sequence is to ensure RPM Command UpdateMetadata is not executed if the TagList in the RPM Command is not included in the Managing SM-DP+ List in the Profile Metadata (rpmConfiguration).

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been loaded on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_ALL\_LPR\_CONF\_NOT\_ALLOWED. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  updateMetadata,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  lprConfiguration { pcmpAddress #TEST\_PCMP\_ADDRESS1 }, NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  updateMetadataResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  commandError  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_LPR\_CONFIG\_OP\_PROF1 | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   iccid #ICCID\_OP\_PROF1  }  }  SW=0x9000 |

Test Sequence #9 Error: RPM Command UpdateMetadata – Allowed CI Public Key Identifier not matched

The purpose of this Test Sequence is to ensure RPM Command UpdateMetadata is not executed if Subject Key Identifier of the CI corresponding to CERT.DPauth.SIG attached to the ongoing session does not match with the Allowed CI Public Key Identifier in the Profile Metadata (rpmConfiguration).

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_OP\_PROF1\_RPM\_CONF\_UM\_CI\_PKI\_RAND is loaded on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  updateMetadata,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  lprConfiguration { pcmpAddress #TEST\_PCMP\_ADDRESS1 },  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  updateMetadataResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  commandError  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_LPR\_CONFIG\_OP\_PROF1) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   iccid #ICCID\_OP\_PROF1  }  }  SW=0x9000 |

Test Sequence #10 Error: RPM Command UpdateMetadata – Case4 - Remove PPR, PPR Update Control Bit set

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_UM\_PPR\_CTRL\_BIT. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  updateMetadata,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  profilePolicyRules {pprUpdateControl, ppr2},  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  updateMetadataResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  pprUpdateInvalidSetting  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PPR\_OP\_PROF1) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   iccid #ICCID\_OP\_PROF1,  profilePolicyRules { pprUpdateControl,ppr1,ppr2 }  }  }  SW=0x9000 |

Test Sequence #11 Error: RPM Command\_UpdateMetadata – Case4 - Invalid RPM Configuration, zero length

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been loaded on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_ALL\_PPR1. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  updateMetadata,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  rpmConfiguration {},  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  updateMetadataResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  invalidRpmConfiguration  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_RPM\_CONFIG\_OP\_PROF1) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   iccid #ICCID\_OP\_PROF1,  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable, disable, delete, listProfileInfo, contactPcmp }  tagList ‘99BA9BBC’H  }  },  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  }  }  SW=0x9000 |

Test Sequence #12 Error: RPM Command\_UpdateMetadata – Case4 - Invalid RPM Comfiguration, Profile Owner OID mismatch

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been loaded on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_ALL\_PPR1. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  updateMetadata,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  rpmConfiguration #RPM\_CONFIG\_OID2,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  updateMetadataResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  invalidRpmConfiguration  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_RPM\_CONFIG\_OP\_PROF1) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   iccid #ICCID\_OP\_PROF1,  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable, disable, delete, listProfileInfo, contactPcmp }  tagList ‘99BA9BBC’H  }  },  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  }  }  SW=0x9000 |

Test Sequence #13 Error: VOID

Test Sequence #14 Error: VOID

Test Sequence #15 Error: VOID

##### 4.2.28.2.6 TC\_eUICC\_ES10b.LoadRPMPackage\_ContactPCMP

Test Sequence #01 Nominal: RPM Command ContactPCMP\_without DPI

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_PCMP. |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  contactPcmp,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  contactPcmpResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  0, -- OK response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  TEST\_PCMP\_ADDRESS1,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |

Test Sequence #02 Nominal: RPM Command ContactPCMP\_with dpiRpm

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_PCMP. |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  contactPcmp,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  #DPI\_RPM1  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  contactPcmpResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  0, -- OK response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  TEST\_PCMP\_ADDRESS1,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |

Test Sequence #03 Error: RPM Command ContactPCMP – ICCID unknown

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_PCMP. |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |
| eUICC | The Operational Profile identified by the ICCID #ICCID\_OP\_PROFX is not loaded. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  contactPcmp,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROFX,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  contactPcmpResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROFX,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  commandError  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |

Test Sequence #04 Error: RPM Command ContactPCMP – SM-DP+ OID not in Managing SM-DP+ List

The purpose of this Test Secuence is to ensure RPM Command ContactPCMP is not executed if the SM-DP+ that sent the RPM Command is not included in the Managing SM-DP+ List in the Profile Metadata (rpmConfiguration).

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC with # METADATA\_OP\_PROF1\_RPM\_CONF\_ALL\_DP\_OID2. |
| eUICC | The PROFILE\_OPERATIONAL1 is Enabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT (  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  contactPcmp,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  contactPcmpResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  commandError  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |

Test Sequence #05 Error: RPM Command ContactPCMP – Allowed CI Public Key Identifier not matched

The purpose of this test sequence is to ensure RPM Command ContactPCMP is not executed if Subject Key Identifier of the CI corresponding to CERT.DPauth.SIG attached to the ongoing session does not match with the Allowed CI Public Key Identifier in the Profile Metadata (rpmConfiguration).

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_OP\_PROF1\_RPM\_CONF\_CP\_CI\_PKI\_RAND is loaded on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  contactPcmp,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  contactPcmpResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  commandError  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |

Test Sequence #06 Error: RPM Command ContactPCMP – Profile is not in Enabled state

The purpose of this Test Sequence is to ensure RPM Command ContactPCMP is not executed if the target Profile is not in Enabled state.

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been loaded on the eUICC with  #METADATA\_OP\_PROF1\_RPM\_CONF\_PCMP. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT (  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  contactPcmp,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  contactPcmpResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  profileNotEnabled  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |

Test Sequence #07 Error: RPM Command ContactPCMP – PCMP Address not configured in Metadata

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been loaded on the eUICC with  #METADATA\_OP\_PROF1\_RPM\_CONF\_NO\_PCMP\_ADDR and it is in Enabled state. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT (  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  contactPcmp,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  contactPcmpResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  noLprConfiguration  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |

##### 4.2.28.2.7 TC\_eUICC\_ES10b.LoadRPMPackage – Multiple RPM Commands

Test Sequence #01 Nominal: RPM Commands Disable, List Profile Info and Delete, profiles of same profile owner

|  |  |
| --- | --- |
| Initial Conditions | |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_OP\_PROF1\_RPM\_CONF\_ALL (without PPR1 and PPR2 present) is loaded and Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 with #METADATA\_OP\_PROF2\_RPM\_CONF\_ALL\_NO\_ENTERP\_CONF (without PPR1 present) is loaded on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_MULT\_CMNDS (  {  MTD\_REQ\_RPM\_CMND (  disable, #ICCID\_OP\_PROF1,  TRUE, NO\_PARAM ),  MTD\_REQ\_RPM\_CMND (  listProfileInfo, #ICCID\_OP\_PROF1,  TRUE, NO\_PARAM ),  MTD\_REQ\_RPM\_CMND (  delete, #ICCID\_OP\_PROF1,  TRUE, NO\_PARAM )  }  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x91XX with the response data MTD\_RES\_RPR\_FOR\_MULT\_CMNDS (  {  MTD\_RES\_RPR\_CMND\_RESULT (  0, disableResult, #ICCID\_OP\_PROF1,  NO\_PARAM, NO\_PARAM),  MTD\_RES\_RPR\_CMND\_RESULT (  1, listProfileInfoResult, NO\_PARAM,  NO\_PARAM,  profileChangeOngoing), MTD\_RES\_RPR\_CMND\_RESULT (  0, deleteResult, #ICCID\_OP\_PROF1,  NO\_PARAM, NO\_PARAM )  }  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_Device 🡪eUICC | FETCH 'XX' | REFRESH Command  (“UICC Reset”) |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  NO\_PARAM,  NO\_PARAM)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO2  }  SW=0x9000 |

Test Sequence #02 Nominal: RPM Commands Enable, List Profile Info and Delete, profiles of different profile owner

|  |  |
| --- | --- |
| Initial Conditions | |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_OP\_PROF1\_RPM\_CONF\_ALL (without PPR1 present) is loaded on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 with #METADATA\_OP\_PROF2\_RPM\_CONF\_ALL\_OWNER2 (without PPR2 present) is loaded on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_MULT\_CMNDS (  {  MTD\_REQ\_RPM\_CMND (  enable, #ICCID\_OP\_PROF2,  TRUE, NO\_PARAM ),  MTD\_REQ\_RPM\_CMND (  listProfileInfo, #ICCID\_OP\_PROF2,  TRUE, NO\_PARAM ),  MTD\_REQ\_RPM\_CMND (  delete, #ICCID\_OP\_PROF1,  TRUE, NO\_PARAM )  )  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x91XX with the response data MTD\_RES\_RPR\_FOR\_MULT\_CMNDS (  {  MTD\_RES\_RPR\_CMND\_RESULT (  0, enableResult, #ICCID\_OP\_PROF2,  NO\_PARAM, NO\_PARAM ), MTD\_RES\_RPR\_CMND\_RESULT (  1, listProfileInfoResult, NO\_PARAM,  NO\_PARAM,  profileChangeOngoing),  MTD\_RES\_RPR\_CMND\_RESULT (  0, deleteResult, #ICCID\_OP\_PROF1,  NO\_PARAM, NO\_PARAM )  }  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_Device 🡪eUICC | FETCH 'XX' | REFRESH Command  (“UICC Reset”) |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  NO\_PARAM,  NO\_PARAM) ) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO2\_ENABLED  }  SW=0x9000 |

Test Sequence #03 Nominal: RPM Commands UpdateMetadata, Enable, List Profile Info, Delete and ContactPCMP, profiles of different profile owner

|  |  |
| --- | --- |
| Initial Conditions | |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_OP\_PROF1\_RPM\_CONF\_ALL\_PPR1 (with PPR1 present) is loaded on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 with #METADATA\_OP\_PROF2\_RPM\_CONF\_ALL\_OWNER2 (without PPR2 present) is loaded on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_MULT\_CMNDS (  {  MTD\_REQ\_RPM\_CMND (  updateMetadata, #ICCID\_OP\_PROF1,  TRUE,{ profilePolicyRules { } } ),  MTD\_REQ\_RPM\_CMND (  enable, #ICCID\_OP\_PROF2, TRUE,  NO\_PARAM ),  MTD\_REQ\_RPM\_CMND (  listProfileInfo, #ICCID\_OP\_PROF2, TRUE,  NO\_PARAM ),  MTD\_REQ\_RPM\_CMND (  delete, #ICCID\_OP\_PROF1, TRUE,  NO\_PARAM ),  MTD\_REQ\_RPM\_CMND (  contactPcmp, #ICCID\_OP\_PROF1,  TRUE, NO\_PARAM )  )  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x91XX with the response data MTD\_RES\_RPR\_FOR\_MULT\_CMNDS (  {  MTD\_RES\_RPR\_CMND\_RESULT (  0, updateMetadataResult, #ICCID\_OP\_PROF1,  NO\_PARAM, NO\_PARAM ),  MTD\_RES\_RPR\_CMND\_RESULT (  0, enableResult, #ICCID\_OP\_PROF2,  NO\_PARAM, NO\_PARAM ),  MTD\_RES\_RPR\_CMND\_RESULT (  1, listProfileInfoResult, NO\_PARAM,  NO\_PARAM,  profileChangeOngoing), MTD\_RES\_RPR\_CMND\_RESULT (  0, deleteResult, #ICCID\_OP\_PROF1,  NO\_PARAM, NO\_PARAM ),  MTD\_RES\_RPR\_CMND\_RESULT (  1, contactPcmp Result, #ICCID\_OP\_PROF1,  NO\_PARAM, commandError)  }  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_Device 🡪eUICC | FETCH 'XX' | REFRESH Command  (“UICC Reset”) |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  NO\_PARAM,  NO\_PARAM) ) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO2\_ENABLED  }  SW=0x9000 |

Test Sequence #04 Nominal: RPM Commands List Profile Info, Disable, Delete and ContactPCMP - Disable command failure

|  |  |
| --- | --- |
| Initial Conditions | |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_OP\_PROF1\_RPM\_CONF\_ALL\_PPR1 (with PPR1 present) is loaded and Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 with #METADATA\_OP\_PROF2\_RPM\_CONF\_ALL\_OWNER2 (without PPR2 present) is loaded on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_MULT\_CMNDS (  {  MTD\_REQ\_RPM\_CMND (  listProfileInfo, #ICCID\_OP\_PROF1, TRUE,  NO\_PARAM ),  MTD\_REQ\_RPM\_CMND (  disable, #ICCID\_OP\_PROF1, TRUE,  NO\_PARAM ),  MTD\_REQ\_RPM\_CMND (  delete, #ICCID\_OP\_PROF2, TRUE,  NO\_PARAM ),  MTD\_REQ\_RPM\_CMND (  contactPcmp, #ICCID\_OP\_PROF1,  TRUE, NO\_PARAM )  )  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_MULT\_CMNDS (  {  MTD\_RES\_RPR\_CMND\_RESULT (  0, listProfileInfoResult, NO\_PARAM,  { #PROFILE\_INFO1 },  NO\_PARAM ),  MTD\_RES\_RPR\_CMND\_RESULT (  1, disableResult, #ICCID\_OP\_PROF1,  NO\_PARAM, disallowedByPolicy),  MTD\_RES\_RPR\_CMND\_RESULT (  0, deleteResult, #ICCID\_OP\_PROF2,  NO\_PARAM, NO\_PARAM ),  MTD\_RES\_RPR\_CMND\_RESULT (  1, contactPcmp Result, #ICCID\_OP\_PROF1,  NO\_PARAM, noLprConfiguration)  }  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  NO\_PARAM,  NO\_PARAM) ) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #05 Nominal: RPM Commands List Profile Info, Disable, Delete and ContactPCMP - Disable command failure and ContinueOnFailure not set

|  |  |
| --- | --- |
| Initial Conditions | |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_OP\_PROF1\_RPM\_CONF\_ALL\_PPR1 (with PPR1 present) is loaded and Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 with #METADATA\_OP\_PROF2\_RPM\_CONF\_ALL\_OWNER2 (without PPR2 present) is loaded on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_MULT\_CMNDS (  {  MTD\_REQ\_RPM\_CMND (  listProfileInfo, #ICCID\_OP\_PROF1, TRUE,  NO\_PARAM ),  MTD\_REQ\_RPM\_CMND (  disable, #ICCID\_OP\_PROF1, NO\_PARAM,  NO\_PARAM ),  MTD\_REQ\_RPM\_CMND (  delete, #ICCID\_OP\_PROF2, TRUE,  NO\_PARAM ),  MTD\_REQ\_RPM\_CMND (  contactPcmp, #ICCID\_OP\_PROF1,  TRUE, NO\_PARAM )  )  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_MULT\_CMNDS (  {  MTD\_RES\_RPR\_CMND\_RESULT (  0, listProfileInfoResult, NO\_PARAM,  { #PROFILE\_INFO1 },  NO\_PARAM ),  MTD\_RES\_RPR\_CMND\_RESULT (  1, disableResult, #ICCID\_OP\_PROF1,  NO\_PARAM, disallowedByPolicy),  }  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2. | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  NO\_PARAM,  NO\_PARAM) ) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1,  #PROFILE\_INFO2  }  SW=0x9000 |

Test Sequence #06 Error: VOID

Test Sequence #07 Error: RPM Commands command after ContactPCMP

|  |  |
| --- | --- |
| Initial Conditions | |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_OP\_PROF1\_RPM\_CONF\_ALL (without PPR1 present) is loaded and Enabled on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL2 with #METADATA\_OP\_PROF2\_RPM\_CONF\_ALL\_OWNER2 (without PPR2 present) is loaded on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_MULT\_CMNDS (  {  MTD\_REQ\_RPM\_CMND (  enable, #ICCID\_OP\_PROF2, TRUE,  NO\_PARAM ),  MTD\_REQ\_RPM\_CMND (  listProfileInfo, #ICCID\_OP\_PROF2, TRUE,  NO\_PARAM ),  MTD\_REQ\_RPM\_CMND (  contactPcmp, #ICCID\_OP\_PROF1,  TRUE, NO\_PARAM )  MTD\_REQ\_RPM\_CMND (  delete, #ICCID\_OP\_PROF1, TRUE,  NO\_PARAM ),  )  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_MULT\_CMNDS (  {  MTD\_RES\_RPR\_CMND\_RESULT (  1, enableResult,  #ICCID\_OP\_PROF2,  NO\_PARAM,  commandError),  MTD\_RES\_RPR\_CMND\_RESULT (  0, listProfileInfoResult,  NO\_PARAM,  { #PROFILE\_INFO2 },  NO\_PARAM ),  MTD\_RES\_RPR\_CMND\_RESULT (  0, contactPcmp Result,  #ICCID\_OP\_PROF1,  NO\_PARAM,  NO\_PARAM)  MTD\_RES\_RPR\_CMND\_RESULT (  2, NO\_PARAM, NO\_PARAM, NO\_PARAM,  commandAfterContactPcmp )  }  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO (  NO\_PARAM,  NO\_PARAM) ) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1,  #PROFILE\_INFO2  }  SW=0x9000 |

##### 4.2.28.2.8 TC\_eUICC\_ES10b.LoadRPMPackage\_ErrorCases

[VOID]

##### 4.2.28.2.9 TC\_eUICC\_ES10b.LoadRPMPackage\_Enterprise\_Profiles

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is loaded on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled. |
| eUICC | The communication between the S\_Device and the eUICC has been initialized by using the [TERMINAL\_CAPABILITY\_Enterprise] and the S\_LPAd has selected the ISD-R.  Common Mutual Authentication procedure has been successfully executed between the eUICC and the S\_SM-DP+ with the operationType is set rpm(1) and enterpriseCapableDevice is present in DeviceInfo (in CtxParams1):   * #GET\_EUICC\_INFO1, #GET\_EUICC\_CHALLENGE and #AUTHENTICATE\_SMDP\_RPM have been sent to the eUICC * the same GSMA CI has been chosen for signing and for verification |

Test Sequence #1 Nominal: RPM Command EnableProfile – Reference Enterprise Profile

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_EN\_REF\_ENTERPRISE\_PROF. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  enable,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x91XX with the response data  MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  enableResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  0, -- OK response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_Device → eUICC | FETCH 'XX' | REFRESH Command  (“UICC Reset”) |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_Enterprise | | |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 6 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 7 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

Test Sequence #2 Error: RPM Command EnableProfile – Other Enterprise Profile, Reference Enterprise Profile is already enabled

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed with #METADATA\_OP\_PROF2\_RPM\_CONF\_EN\_REF\_ENTERPRISE\_PROF before the PROFILE\_OPERATIONAL1 and is Enabled on the eUICC.  (PPR1 is not set) |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_EN\_OTHER\_ENTERPRISE\_PROF. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  enable,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x91XX with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  enableResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  commandError  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_Enterprise | | |
| 3 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA  (#GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  #PROFILE\_INFO2\_ENABLED  }  SW=0x9000 |
| 5 | S\_Device → eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 6 | S\_Device → eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF2  SW=0x9000 |

Test Sequence #03 Nominal: RPM Command EnableProfile - target profile is the Reference Enterprise profile, only this profile can be enabled

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_REF\_PROF\_ONLY\_THIS\_EN |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  enable,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x91XX with the response data  MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  enableResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  0, -- OK response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_Device 🡪eUICC | FETCH 'XX' | REFRESH Command  (“UICC Reset”) |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_Enterprise | | |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  MTD\_GET\_PROFILE\_INFO(  #ICCID\_OP\_PROF1,  NO\_PARAM)) | response ProfileInfoListResponse ::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |
| 6 | S\_Device 🡪 eUICC | [SELECT\_ICCID] | SW=0x9000 |
| 7 | S\_Device 🡪 eUICC | [READ\_BINARY] with <L>=0x0A | #ICCID\_OP\_PROF1  SW=0x9000 |

Test Sequence #4 Nominal: RPM Command UpdateMetadata – Enterprise Configuration - set referenceEnterpriseRule bit - profile is enabled

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_EN\_OTHER\_ENTERPRISE\_PROF and is Enabled on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  updateMetadata,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  enterpriseConfiguration #ENTERPRISE\_CONFIG1,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  updateMetadataResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  0, -- OK response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  #GET\_ENTERPRISE\_CONFIG\_ OP\_PROF1) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   iccid #ICCID\_OP\_PROF1,  enterpriseConfiguration #ENTERPRISE\_CONFIG1  }  }  SW=0x9000 |

Test Sequence #05 Nominal: RPM Command UpdateMetadata – Enterprise Configuration - set referenceEnterpriseRule bit for another enterprise profile

The purpose of this test sequence is to verify if the eUICC unset the referenceEnterpriseRule bit of the Enterprise Profile for which it is currently set when an RPM command is executed to update the target enterprise profile with the referenceEnterpriseRule bit being set.

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed with #METADATA\_OP\_PROF2\_RPM\_CONF\_EN\_REF\_ENTERPRISE\_PROF on the eUICC . |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed and enabled on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_EN\_OTHER\_ENTERPRISE\_PROF. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  updateMetadata,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  enterpriseConfiguration #ENTERPRISE\_CONFIG2,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  updateMetadataResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  0, -- OK response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  #GET\_ENTERPRISE\_CONFIG\_ OP\_PROF1) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   iccid #ICCID\_OP\_PROF1,  enterpriseConfiguration #ENTERPRISE\_CONFIG2  }  }  SW=0x9000 |
| 3 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  #GET\_ENTERPRISE\_CONFIG\_ OP\_PROF2) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   iccid #ICCID\_OP\_PROF2,  enterpriseConfiguration {  enterpriseOid #S\_ENTERPRISE\_OID,  enterpriseName #ENTERPRISE\_NAME1,  enterpriseRules {  enterpriseRuleBits{  priorityEnterpriseProfile, onlyEnterpriseProfilesCanBeInstalled  },  numberOfNonEnterpriseProfiles 0  }  }  }  }  SW=0x9000 |

Test Sequence #06 Nominal: RPM Command UpdateMetadata – Enterprise Configuration - Rule 1

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed and enabled on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_ENTERPRISE\_REF\_RULE3 |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  updateMetadata,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  enterpriseConfiguration #ENTERPRISE\_CONFIG1,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  updateMetadataResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  0, -- OK response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  #GET\_ENTERPRISE\_CONFIG\_ OP\_PROF1) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   iccid #ICCID\_OP\_PROF1,  enterpriseConfiguration #ENTERPRISE\_CONFIG1  }  }  SW=0x9000 |

Test Sequence #07 Nominal: RPM Command UpdateMetadata – Enterprise Configuration - Rule 2

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been installed and enabled on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_ENTERPRISE\_REF\_RULE3 |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  updateMetadata,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  enterpriseConfiguration #ENTERPRISE\_CONFIG2,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  updateMetadataResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  0, -- OK response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA(  #GET\_ENTERPRISE\_CONFIG\_ OP\_PROF1) | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {   iccid #ICCID\_OP\_PROF1,  enterpriseConfiguration #ENTERPRISE\_CONFIG2  }  }  SW=0x9000 |

Test Sequence #08 Nominal: RPM Command\_ListProfileInfo\_- ICCID specified and tagList with enterprise config tag

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed and enabled on the eUICC with #METADATA\_OP\_PROF2\_RPM\_CONF\_ALL\_ENTERP\_RULES |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_FOR\_LIST\_PROFILE\_INFO (  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF2,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  ‘5ABD9F1F’H  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  listProfileInfoResult,  <S\_TRANSACTION\_ID>,  NO\_PARAM,  0, -- OK response  {  #NOTIF\_METADATA\_PROF1\_DP1\_RPR  },  #S\_SM\_DP+\_OID,  { PROFILES\_INFO\_RPM\_TAGLIST6 },  NO\_PARAM,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |

Test Sequence #09 Nominal: RPM Command Enable - Other Enterprise profile on a non-enterprise device

The purpose of this test sequence is to verify that an Other Enterprise profile without Enterprise Rules can be enabled on a non-enterprise device using RPM command.

|  |  |
| --- | --- |
| Initial Conditions | |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been loaded on the eUICC with  #METADATA\_OP\_PROF1\_RPM\_CONF\_EN\_OTHER\_ENTERPRISE\_PROF\_UM\_ENT\_CONF in a non-Enterprise S\_Device |
| eUICC | The PROFILE\_OPERATIONAL2 that is not an enterprise profile has been installed on the eUICC with #METADATA\_OP\_PROF2\_RPM\_CONF\_EN. |
| eUICC | The communication between the S\_Device (non-Enterprise S\_Device) and the eUICC has been initialized and the S\_LPAd has selected the ISD-R.  Common Mutual Authentication procedure has been successfully executed between the eUICC and the S\_SM-DP+:  · #GET\_EUICC\_INFO1, #GET\_EUICC\_CHALLENGE and #AUTHENTICATE\_SMDP\_RPM have been sent to the eUICC  the same GSMA CI has been chosen for signing and for verification |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  enable,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x91XX with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  enableResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  0, -- OK response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  NO\_PARAM  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_Device 🡪eUICC | FETCH 'XX' | REFRESH Command |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| 4 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 5 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA  (#GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  #PROFILE\_INFO2  }  SW=0x9000 |

Test Sequence #10 Error: RPM Command UpdateMetadata - Enterprise Configuration with Enterprise Rules on an Enterprise Profile without Enterprise Rules

The purpose of this test sequence is to verify that eUICC returns enterpriseConfigurationNotAllowed when RPM command tries to update Enterprise Configuration with Enterprise Rules on an Enterprise Profile does not already contain Enterprise Rules.

|  |  |
| --- | --- |
| Initial Conditions | |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 has been loaded on the eUICC and enabled with  #METADATA\_OP\_PROF1\_RPM\_CONF\_EN\_OTHER\_ENTERPRISE\_PROF\_UM\_ENT\_CONF |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT( MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  updateMetadata,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  enterpriseConfiguration #ENTERPRISE\_CONFIG2,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  updateMetadataResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  enterpriseConfigurationNotAllowed  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA  (#GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  }  SW=0x9000 |

Test Sequence #11 Error: RPM Command EnableProfile – non-Enterprise Profile, Reference Enterprise Profile is already enabled with Rule 2

The purpose of this test sequence is to verify if eUICC returns the error enterpriseRulesNotAllowed when attempting to enable a Profile via RPM command and if the eUICC contains an enabled Profile with a Reference Enterprise Rule indicating "Only Enterprise Profiles can be enabled" (Rule 2) and the target Profile is not an Enterprise Profile.

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed with  #METADATA\_OP\_PROF2\_RPM\_CONF\_ALLOW\_NON\_ENTERPRISE\_PROF\_INS before the PROFILE\_OPERATIONAL1 on the eUICC.  (PPR1 is not set) |
| eUICC | The PROFILE\_OPERATIONAL1 that is not an enterprise profile has been installed on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_EN. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  enable,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  enableResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  disallowedByEnterpriseRule  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA  (#GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO2,  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #12 Error: RPM Command EnableProfile – Other Enterprise Profile, Reference Enterprise Profile is already enabled with Rule 1

The purpose of this test sequence is to verify if eUICC returns the error enterpriseRulesNotAllowed when attempting to enable another Enterprise Profile via RPM command and if the eUICC contains an enabled Profile with a Reference Enterprise Rule indicating "Only this Profile can be enabled" (Rule 1).

|  |
| --- |
| Initial Conditions |
| Entity | Description of the initial condition |
| eUICC | The PROFILE\_OPERATIONAL2 has been installed with #METADATA\_OP\_PROF2\_RPM\_CONF\_EN\_ONLY\_THIS\_REF\_ENTERPRISE\_PROF before the PROFILE\_OPERATIONAL1 on the eUICC.  (PPR1 is not set) |
| eUICC | The PROFILE\_OPERATIONAL1 another enterprise profile has been installed on the eUICC with #METADATA\_OP\_PROF1\_RPM\_CONF\_EN\_OTHER\_ENTERPRISE\_PROF. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  enable,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM  )  ) | SW=0x9000 without response data for all STORE DATA commands except for the last one  SW=0x9000 with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  enableResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF1,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  disallowedByEnterpriseRule  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_LPAd 🡪eUICC | MTD\_STORE\_DATA  (#GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO2,  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |

Test Sequence #13 Error: VOID

##### 4.2.28.2.10 TC\_eUICC\_ES10b.LoadRPMPackage\_CatBusy

Test Sequence #01 Error: RPM Command EnableProfile – Cat busy

The purpose of this Test Sequence is to ensure RPM Command EnableProfile is not executed if there is an active proactive session.

|  |
| --- |
| Initial Conditions |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL2 has been loaded on the eUICC with  #METADATA\_OP\_PROF2\_RPM\_CONF\_EN. |
| eUICC | The PROFILE\_OPERATIONAL2 is in Disabled state. |
| eUICC | The PROFILE\_OPERATIONAL1 has been loaded with #METADATA\_OP\_PROF1\_RPM\_CONF\_EN on the eUICC and is Enabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [GET\_MNO\_SD]) | SW=0x91XX |
| IC2 | Do not send FETCH command | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT (  MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND (  enable,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF2,  <S\_SM\_DP+\_SIGNATURE3>,  NO\_PARAM,  NO\_PARAM,  NO\_PARAM  )  ) | SW= 0x91XX without response data for all STORE DATA commands except for the last one  SW=0x91XX with the response data MTD\_RES\_RPR\_FOR\_SINGLE\_CMND  (  enableResult,  <S\_TRANSACTION\_ID>,  #ICCID\_OP\_PROF2,  1, -- error response  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #S\_SM\_DP+\_OID,  NO\_PARAM,  NO\_PARAM,  catBusy  )  for the last STORE DATA command  • Verify the euiccSignRPR <EUICC\_SIGN\_RPR> using the #PK\_EUICC\_SIG |
| 2 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x00 – POR OK |
| 3 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1  #PROFILE\_INFO2  }  SW=0x9000 |

## 4.3 VOID

## 4.4 VOID

## 4.5 VOID

## 4.6 VOID

## 4.7 LPAe Interfaces

This section is defined as FFS.

# 5 Procedure - Behaviour Testing

## 5.1 General Overview

This section focuses on the functional behaviour of the eUICC according to the GSMA RSP Technical Specification [2].

## 5.2 eUICC Behaviour

### 5.2.1 Retry mechanism

#### 5.2.1.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 2.6.7.1
* Section 3.1.3.2
* Section 5.7.5, 5.7.7, 5.7.14

#### 5.2.1.2 Test Cases

##### 5.2.1.2.1 TC\_eUICC\_PrepareDownload\_Retry\_ReuseOTKeys

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is not loaded on the eUICC. |
| eUICC | The communication between the S\_Device and the eUICC has been initialized and the S\_LPAd has selected the ISD-R.  Common Mutual Authentication procedure has been successfully executed between the eUICC and the S\_SM-DP+ |

Test Sequence #01 Nominal: Confirmation Code retry mechanism by reusing previous One-Time key pair

The purpose of this test is to verify the Confirmation Code retry mechanism. The S\_LPAd simulates that an incorrect Confirmation Code has been filled by the End User. Then, the S\_LPAd sends another ES10b.PrepareDownload function with a correct Confirmation Code value. In this case, the eUICC does not have to generate a new one-time key pair and uses the previous one given by the SM-DP+.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step | Direction | | Sequence / Description | Expected result |
| IC1 | <S\_HASHED\_CC> = MTD\_GENERATE\_HASHED\_CC(#CONFIRMATION\_CODE1, <S\_TRANSACTION\_ID>) | | | |
| 1 | S\_LPAd → eUICC | | MTD\_STORE\_DATA\_SCRIPT(  #PREP\_DOWNLOAD\_WITH\_CC) | #R\_PREP\_DOWNLOAD\_WITH\_CC  SW=0x9000  The <EUICC\_SIGNATURE2> SHALL be verified with the #PK\_EUICC\_SIG.  Verify that the <S\_TRANSACTION\_ID> present in the euiccSigned2 is the same as in #PREP\_DOWNLOAD\_WITH\_CC.  Verify that the <S\_HASHED\_CC> present in the euiccSigned2 is the same as in #PREP\_DOWNLOAD\_WITH\_CC.  Extract the <OTPK\_EUICC\_ECKA> and reuse the same value in step 4 |
| 2 | Execute the Common Mutual Authentication procedure between the eUICC and the S\_SM-DP+   * the same GSMA CI as for the first attempt has been chosen for signing and for verification | | | |
| 3 | <S\_HASHED\_CC> = MTD\_GENERATE\_HASHED\_CC(#CONFIRMATION\_CODE2, <S\_TRANSACTION\_ID>) | | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  #PREP\_DOWNLOAD\_RETRY\_CC) | | #R\_PREP\_DOWNLOAD\_WITH\_CC  SW=0x9000  The <EUICC\_SIGNATURE2> SHALL be verified with the #PK\_EUICC\_SIG.  Verify that the <S\_TRANSACTION\_ID> present in the euiccSigned2 is the same as in #PREP\_DOWNLOAD\_RETRY\_CC.  Verify that the <S\_HASHED\_CC> present in the euiccSigned2 is the same as in #PREP\_DOWNLOAD\_RETRY\_CC.  Verify that the <OTPK\_EUICC\_ECKA> present in the euiccSigned2 is the same as in #PREP\_DOWNLOAD\_RETRY\_CC. |

Test Sequence #02 Nominal: Retry after a CancelSession Reason “Postponed”

The purpose of this test is to verify that the eUICC can reuse the one-time key pair generated during a previous attempt. In this case, the S\_LPAd simulates that the End User has postponed the download of the Profile.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | <S\_HASHED\_CC> = MTD\_GENERATE\_HASHED\_CC(#CONFIRMATION\_CODE1, <S\_TRANSACTION\_ID>) | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  #PREP\_DOWNLOAD\_WITH\_CC) | #R\_PREP\_DOWNLOAD\_WITH\_CC  SW=0x9000  The <EUICC\_SIGNATURE2> SHALL be verified with the #PK\_EUICC\_SIG.  Verify that the <S\_TRANSACTION\_ID> present in the euiccSigned2 is the same as in #PREP\_DOWNLOAD\_WITH\_CC.  Verify that the <S\_HASHED\_CC> present in the euiccSigned2 is the same as in #PREP\_DOWNLOAD\_WITH\_CC.  Extract the <OTPK\_EUICC\_ECKA> and reuse the same value in step 4 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA( #CANCEL\_SESSION\_POSTPONED) | #R\_CANCEL\_SESSION\_POSTPONED  SW = 0x9000 |
| 3 | Execute the Common Mutual Authentication procedure between the eUICC and the S\_SM-DP+   the same GSMA CI as for the first attempt has been chosen for signing and for verification | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  #PREP\_DOWNLOAD\_RETRY\_C) | #R\_PREP\_DOWNLOAD\_WITH\_CC  SW=0x9000  The <EUICC\_SIGNATURE2> SHALL be verified with the #PK\_EUICC\_SIG.  Verify that the <S\_TRANSACTION\_ID> present in the euiccSigned2 is the same as in #PREP\_DOWNLOAD\_RETRY\_CC.  Verify that the <S\_HASHED\_CC> present in the euiccSigned2 is the same as in #PREP\_DOWNLOAD\_RETRY\_CC.  Verify that the <OTPK\_EUICC\_ECKA> present in the euiccSigned2 is the same as in #PREP\_DOWNLOAD\_RETRY\_CC. |

Test Sequence #03 Nominal: Retry after a CancelSession Reason “Timeout”

The purpose of this test is to verify that the eUICC can reuse the one-time key pair generated during a previous attempt. In this case, the S\_LPAd simulates that the End User does not confirm the download of the Profile within the timeout interval.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | <S\_HASHED\_CC> = MTD\_GENERATE\_HASHED\_CC(#CONFIRMATION\_CODE1, <S\_TRANSACTION\_ID>) | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  #PREP\_DOWNLOAD\_WITH\_CC) | #R\_PREP\_DOWNLOAD\_WITH\_CC  SW=0x9000  The <EUICC\_SIGNATURE2> SHALL be verified with the #PK\_EUICC\_SIG.  Verify that the <S\_TRANSACTION\_ID> present in the euiccSigned2 is the same as in #PREP\_DOWNLOAD\_WITH\_CC.  Verify that the <S\_HASHED\_CC> present in the euiccSigned2 is the same as in #PREP\_DOWNLOAD\_WITH\_CC.  Extract the <OTPK\_EUICC\_ECKA> and reuse the same value in step 4 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #CANCEL\_SESSION\_TIMEOUT) | #R\_CANCEL\_SESSION\_TIMEOUT  SW = 0x9000 |
| 3 | Execute the Common Mutual Authentication procedure between the eUICC and the S\_SM-DP+   the same GSMA CI as for the first attempt has been chosen for signing and for verification | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  #PREP\_DOWNLOAD\_RETRY\_CC) | #R\_PREP\_DOWNLOAD\_WITH\_CC  SW=0x9000  The <EUICC\_SIGNATURE2> SHALL be verified with the #PK\_EUICC\_SIG.  Verify that the <S\_TRANSACTION\_ID> present in the euiccSigned2 is the same as in #PREP\_DOWNLOAD\_RETRY\_CC.  Verify that the <S\_HASHED\_CC> present in the euiccSigned2 is the same as in #PREP\_DOWNLOAD\_RETRY\_CC.  Verify that the <OTPK\_EUICC\_ECKA> present in the euiccSigned2 is the same as in #PREP\_DOWNLOAD\_RETRY\_CC. |

##### 5.2.1.2.2 TC\_eUICC\_PrepareDownload\_Retry\_NewOTKeys

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is not loaded on the eUICC |
| eUICC | The communication between the S\_Device and the eUICC has been initialized and the S\_LPAd has selected the ISD-R.  Common Mutual Authentication procedure has been successfully executed between the eUICC and the S\_SM-DP+ |

Test Sequence #01 Nominal: Confirmation Code retry mechanism by not reusing previous One-Time key pair

The purpose of this test is to verify the Confirmation Code retry mechanism. The S\_LPAd simulates that an incorrect Confirmation Code has been filled by the End User. Then, the S\_LPAd sends another ES10b.PrepareDownload function with a correct Confirmation Code value. In this case, the eUICC does not support the storage of unused one-time key pair or the eUICC has discarded the previous one-time public key: we expect the eUICC to generate a new set of keys.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | <S\_HASHED\_CC> = MTD\_GENERATE\_HASHED\_CC(#CONFIRMATION\_CODE1, <S\_TRANSACTION\_ID>) | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  #PREP\_DOWNLOAD\_WITH\_CC) | #R\_PREP\_DOWNLOAD\_WITH\_CC  SW=0x9000  The <EUICC\_SIGNATURE2> SHALL be verified with the #PK\_EUICC\_SIG.  Verify that the <S\_TRANSACTION\_ID> present in the euiccSigned2 is the same as in #PREP\_DOWNLOAD\_WITH\_CC.  Verify that the <S\_HASHED\_CC> present in the euiccSigned2 is the same as in #PREP\_DOWNLOAD\_WITH\_CC.  Extract the <OTPK\_EUICC\_ECKA> and reuse the same value in step 4 |
| 2 | Execute the Common Mutual Authentication procedure between the eUICC and the S\_SM-DP+   * the same GSMA CI as for the first attempt has been chosen for signing and for verification | | |
| 3 | <S\_HASHED\_CC> = MTD\_GENERATE\_HASHED\_CC(#CONFIRMATION\_CODE2, <S\_TRANSACTION\_ID>) | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  #PREP\_DOWNLOAD\_RETRY\_CC) | #R\_PREP\_DOWNLOAD\_WITH\_CC  SW=0x9000  The <EUICC\_SIGNATURE2> SHALL be verified with the #PK\_EUICC\_SIG.  Verify that the <S\_TRANSACTION\_ID> present in the euiccSigned2 is the same as in #PREP\_DOWNLOAD\_RETRY\_CC.  Verify that the <S\_HASHED\_CC> present in the euiccSigned2 is the same as in #PREP\_DOWNLOAD\_RETRY\_CC.  Verify that the <OTPK\_EUICC\_ECKA> present in the euiccSigned2 is NOT the same as in #PREP\_DOWNLOAD\_RETRY\_CC. |

### 5.2.2 Forbidden PPRs

#### 5.2.2.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 2.5.6.1
* Section 4.3
* Section 5.5.3

#### 5.2.2.2 Test Cases

##### 5.2.2.2.1 TC\_eUICC\_ForbiddenPPRs

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | There is no Profile installed in the eUICC. |

Test Sequence #01 Nominal: PPR1 management and handling when Operational Profile is installed

The purpose of this test is to verify that the eUICC automatically sets PPR1 in the forbiddenProfilePolicyRules of EUICCInfo2 when an Operational Profile is installed. Any Operational Profile with PPR1 SHALL be rejected by the eUICC once an Operational Profile has been installed.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_EUICC\_INFO2) | forbiddenProfilePolicyRules in EUICCInfo2 does not contain ppr1 |
| 2 | Install PROFILE\_OPERATIONAL1 | | |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_EUICC\_INFO2) | forbiddenProfilePolicyRules in EUICCInfo2 contains ppr1 |
| 4 | Execute the Common Mutual Authentication procedure between the eUICC and the S\_SM-DP+ | | |
| 5 | Execute the Sub-procedure Profile Download and Installation – End User Confirmation between the eUICC and the S\_SM-DP+   #PREP\_DOWNLOAD\_NO\_CC is sent to the eUICC | | |
| 6 | Generate the <OTPK\_S\_SM\_DP+\_ECKA> and <OT\_SK\_S\_SM\_DP+\_ECKA> | | |
| 7 | <BPP> = MTD\_GENERATE\_BPP(  #S\_INIT\_SC\_PROF1,  #CONF\_ISDP\_EMPTY,  #METADATA\_OP\_PROF4,  NO\_PARAM,  #UPP\_OP\_PROF4 | | |
| 8 | Split the <BPP> into several segments arrays named:   <BPP\_SEG\_INIT>   <BPP\_SEG\_A0>   <BPP\_SEG\_A1>   <BPP\_SEG\_A3> | | |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_INIT>) | SW=0x9000 without response data for all STORE DATA commands |
| 10 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A0>) | SW=0x9000 without response data for all STORE DATA commands |
| 11 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  <BPP\_SEG\_A1>) | SW=0x9000 with the response data #R\_PIR\_PPR\_NOT\_ALLOWED |
| 12 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk : {  #PROFILE\_INFO1\_DISABLED  }  SW=0x9000 |
| 13 | Delete PROFILE\_OPERATIONAL1 | | |
| 14 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_EUICC\_INFO2) | forbiddenProfilePolicyRules in EUICCInfo2 does not contain ppr1 |

### 5.2.3 eUICC's RAT

#### 5.2.3.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 3.1.3.2
* Section 3.2.4
* Section 5.7.15, 5.7.22

#### 5.2.3.2 Test Cases

##### 5.2.3.2.1 TC\_eUICC\_GetProfilesInfo\_GetRAT\_RSPSession

Test Sequence #01 Nominal: GetProfilesInfo and GetRAT during RSP session

The purpose of this test is to ensure that the eUICC can be requested during a RSP session context to retrieve the list of installed Profiles and the Rules Authorization Table.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | The eUICC's RAT is configured as defined in section G.2.4:   * one PPAR authorizing PPR1 and PPR2 for all MNOs with End User consent required (i.e. #PPRS\_ALLOWED) * no additional rules |
| eUICC | The PROFILE\_OPERATIONAL1 is installed and Enabled on the eUICC |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_EUICC\_INFO1) | #R\_EUICC\_INFO1  SW = 0x9000  Extract the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING> and <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION> from response data and verify if they contain at least one same GSMA CI Key ID |
| IC4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_EUICC\_CHALLENGE) | #R\_CHALLENGE  SW = 0x9000  Extract the <EUICC\_CHALLENGE> |
| IC5 | The following inputs are required for Step IC6 as described in the InitiateAuthentication function:   * <S\_TRANSACTION\_ID> * <EUICC\_CHALLENGE> * <S\_SMDP\_CHALLENGE> * <S\_SMDP\_SIGNATURE1> * Set the <EUICC\_CI\_PK\_ID\_TO\_BE\_USED> to the CI Key ID in highest priority from the <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING> * Choose the #CERT\_S\_SM\_DPauth\_SIG leading to the same Root CI certificate | | |
| IC6 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  NO\_PARAM,  #CRL\_LIST,  FALSE  )  ) | MTD\_CHECK\_AUTH\_SERVER\_RESP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1)  SW = 0x9000 |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_RAT) | #R\_DEFAULT\_RAT with exact same structure and order  SW = 0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_PROFILES\_INFO\_ALL) | response ProfileInfoListResponse::= profileInfoListOk: {  #PROFILE\_INFO1  }  SW = 0x9000 |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  #PREP\_DOWNLOAD\_NO\_CC) | #R\_PREP\_DOWNLOAD\_NO\_CC  SW=0x9000 |

### 5.2.4 eUICC File Structure

#### 5.2.4.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 3.4.1, 3.4.2, 3.4.3

#### 5.2.4.2 Test Cases

##### 5.2.4.2.1 TC\_eUICC\_Default\_FileSystem

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | There is no Profile installed in the eUICC. |

Test Sequence #01 Nominal: Default file system available

The purpose of this test is to verify that if there is no Profile on the eUICC, the eUICC still ensures a file system to the Device.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_Device → eUICC | RESET | ATR present |
| 2 | S\_Device → eUICC | [SELECT\_MF] | FCP Template present with tag 0xA5 (Proprietary Information) containing 0x87 01 01 (Supported system commands = TERMINAL CAPABILITY)  SW=0x9000 |
| 3 | S\_Device → eUICC | [TERMINAL\_CAPABILITY\_LPAd] | SW=0x9000 |
| 4 | S\_Device → eUICC | [TERMINAL\_PROFILE] | Toolkit initialization THEN SW=0x9000 |

### 5.2.5 eUICC Delete Profile Process

#### 5.2.5.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 2.4.4
* Section 3.0.1, 3.1.4
* Section 5.7.8

#### 5.2.5.2 Test Cases

##### 5.2.5.2.1 TC\_eUICC\_DeleteProfile\_ISDP\_And\_Components

Test Sequence #01 Nominal: ISD-P and Profile Components Deletion

The purpose of this test is to verify that when a Profile is deleted, the eUICC removes the ISD-P and all Profile Components related to it. In order to do so, we are checking the eUICC Non-Volatile Memory variation.

|  |  |
| --- | --- |
| Initial Conditions |  |
| Entity | Description of the initial condition |
| eUICC | There is no Profile installed on the eUICC. |

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_EUICC\_INFO2) | Retrieve free non-volatile memory value (tag 0x82) from <EXT\_CARD\_RESOURCE> in EUICCInfo2 as <FREE\_MEMORY\_NO\_PROFILE> |
| IC4 | Install PROFILE\_OPERATIONAL1 | | |
| IC5 | Remove all Install Notifications from eUICC | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_EUICC\_INFO2) | Retrieve free non-volatile memory value (tag 0x82) from <EXT\_CARD\_RESOURCE> in EUICCInfo2 as <FREE\_MEM\_OP\_PROF1\_INSTALLED>  Verify that <FREE\_MEM\_OP\_PROF1\_INSTALLED> is lower than <FREE\_MEMORY\_NO\_PROFILE> |
| 2 | Delete PROFILE\_OPERATIONAL1 | | |
| 3 | Remove the Delete Notification from eUICC | | |
| 4 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #GET\_EUICC\_INFO2) | Retrieve free non-volatile memory value (tag 0x82) from <EXT\_CARD\_RESOURCE> in EUICCInfo2 as <FREE\_MEM\_OP\_PROF1\_DELETED>  Verify that <FREE\_MEM\_OP\_PROF1\_DELETED> is higher than <FREE\_MEM\_OP\_PROF1\_INSTALLED> |

### 5.2.6 eUICC Enable Profile Process

#### 5.2.6.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 3.5
* Section 5.5.5
* Section 5.7.16

#### 5.2.6.2 Test Cases

##### 5.2.6.2.1 TC\_eUICC\_EnableProfile\_Twice\_Notifications

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 is installed and Enabled on the eUICC. |
| eUICC | No Notification is stored in the eUICC's Pending Notifications List. |

Test Sequence #01 Nominal: Notifications generation

The purpose of this test is to verify that when an Enable Profile operation is performed and the current Enabled Profile is implicitly Disabled, both Notifications are generated. The eUICC automatically increments its sequence number each time a Notification is generated across all Profiles.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Install PROFILE\_OPERATIONAL2  The default Profile downloading procedure defined in section 2.2.3.1 SHALL be used with the following exceptions:   #CERT\_S\_SM\_DP2auth\_SIG SHALL be set in MTD\_AUTHENTICATE\_SMDP rather than #CERT\_S\_SM\_DPauth\_SIG   #TEST\_DP\_ADDRESS2 SHALL be set in MTD\_AUTHENTICATE\_SMDP rather than #TEST\_DP\_ADDRESS1   #CERT\_S\_SM\_DP2pb\_SIG SHALL be set in #PREP\_DOWNLOAD\_NO\_CC rather than #CERT\_S\_SM\_DPpb\_SIG | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | |
| 2 | Remove the ProfileInstallationResult and OtherSignedNotification for Install | | |
| 3 | Enable PROFILE\_OPERATIONAL2 | | |
| 4 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| 5 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DI1\_EN2 SW = 0x9000  Verify that  <NOTIF\_SEQ\_NO\_IN2> is lower than <NOTIF\_SEQ\_NO\_DI1>.  Verify that  <NOTIF\_SEQ\_NO\_DI1> and <NOTIF\_SEQ\_NO\_EN2> follow this order in an incremental sequence |
| NOTE: In order to compare the sequence numbers, the test tool can retrieve the <NOTIF\_SEQ\_NO\_IN2\_PIR> value through the PIR returned at the end of the step IC3. | | | |

### 5.2.7 eUICC Disable Profile Process

#### 5.2.7.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 2.4.5

#### 5.2.7.2 Test Cases

##### 5.2.7.2.1 TC\_eUICC\_DisableProfile\_ApplicationManagement

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | PROFILE\_OPERATIONAL1 is installed and Enabled. |

Test Sequence #01 Nominal: Application Selection/Deletion not available on Disabled Profile

The purpose of this test is to verify that when a Profile is Disabled, the eUICC does not allow the selection or deletion of any application within the Profile.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | S\_Device → eUICC | [SELECT\_USIM] | FCP Template present SW=0x9000 |
| IC3 | S\_Device → eUICC | MTD\_SELECT(  #SSD\_AID) | SSD is selected  SW=0x9000 |
| IC4 | Disable PROFILE\_OPERATIONAL1 | | |
| 1 | S\_Device → eUICC | [SELECT\_USIM] | USIM is not found  SW=0x6A82 |
| 2 | S\_Device → eUICC | MTD\_SELECT(  #SSD\_AID) | SSD is not found  SW=0x6A82 |
| 3 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| 4 | S\_Device → eUICC | MTD\_SEND\_SMS\_PP(  [DELETE\_SSD]) | SW=0x91XX  or SW=0x9000 (i.e. envelope rejected, see Note)  or any error SW (i.e. envelope rejected,see Note) |
| 5 | S\_Device → eUICC | FETCH 'XX' | SMS POR received  SCP80 response status code equal to 0x06 (Unidentified security error) or 0x09 (TAR unknown) |
| 6 | S\_Device → eUICC | TERMINAL RESPONSE | SW=0x9000 |
| 7 | Enable PROFILE\_OPERATIONAL1 | | |
| 8 | S\_Device → eUICC | MTD\_SELECT(  #SSD\_AID) | SSD is selected  SW=0x9000 |
| NOTE: Depending on the implementation, the eUICC MAY decide to not send back a POR (e.g. SW=0x9000 on the ENVELOPE command). Therefore, the steps 5 and 6 SHALL only be executed in case SW=0x91XX. | | | |

### 5.2.8 eUICC Notifications

#### 5.2.8.1 Conformance Requirements

**References**

GSMA RSP Technical Specification [2]:

* Section 5.7.16, 5.7.17, 5.7.18

#### 5.2.8.2 Test Cases

##### 5.2.8.2.1 TC\_eUICC\_Enable\_Disable\_Delete\_Notifications

|  |  |
| --- | --- |
| General Initial Conditions | |
| Entity | Description of the general initial condition |
| eUICC | The PROFILE\_OPERATIONAL1 with #METADATA\_EN\_DI\_DE\_NOTIFS is loaded on the eUICC. |
| eUICC | The PROFILE\_OPERATIONAL1 is Disabled. |
| eUICC | No Notification is stored in the eUICC's Pending Notifications List. |

Test Sequence #01 Nominal: Multiple Enable, Disable and Delete Notifications

The purpose of this test is to verify that when a Local Profile Management Operation (i.e. Enable, Disable and Delete Profile) is performed, all Notifications configured in the notificationConfigurationInfo are generated by the eUICC.

NOTE: In this sequence, the maximum number of Notifications simultaneously tested has been set as to two as there is not minimum defined in SGP.21 or SGP.22 [2] for the number of Notifications that can be stored by the eUICC.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Expected result |
| IC1 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC2 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| IC3 | Enable PROFILE\_OPERATIONAL1 | | |
| IC4 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| IC5 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_EN1\_EN1  SW = 0x9000 |
| 2 | Remove all the pending notifications | | |
| 3 | Disable PROFILE\_OPERATIONAL1 | | |
| 4 | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | | |
| 5 | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | | |
| 6 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DI1\_DI1 SW = 0x9000 |
| 7 | Remove all the pending notifications | | |
| 8 | Delete PROFILE\_OPERATIONAL1 | | |
| 9 | S\_LPAd → eUICC | MTD\_STORE\_DATA(  #LIST\_NOTIF\_ALL) | #R\_LIST\_NOTIF\_DE1\_DE1 SW = 0x9000 |

## 5.3 VOID

## 5.4 VOID

# 6 VOID

# 7 VOID

Annex A Constants

A.1 Generic Constants

|  |  |
| --- | --- |
| Name | Content |
| ACTIVATION\_CODE\_1 | 1$#TEST\_DP\_ADDRESS1$#MATCHING\_ID\_1 |
| ACTIVATION\_CODE\_2 | 1$#TEST\_DP\_ADDRESS1$#MATCHING\_ID\_2$#S\_SM\_DP+\_OID |
| ADDITIONAL\_SMDP\_DATA\_EXCEEDED\_MAX | 0x01 02 03…76 77 78  -- additional data objects defined by the S\_SM-DP+ depending on the length of the SM-DP+ OID, to ensure that total length of dpProprietaryData is 129 bytes |
| ADDITIONAL\_SMDP\_DATA\_MAX\_LENGTH | 0x01 02 03…75 76 77  -- additional data objects defined by the S\_SM-DP+ depending on the length of the SM-DP+ OID, to ensure that total length of dpProprietaryData is 128 bytes |
| CONFIRMATION\_CODE1 | 0102030405 |
| CONFIRMATION\_CODE2 | ABCDEFGHIJ |
| CTX\_PARAMS1  (CtxParams1) | ctxParamsForCommonAuthentication : {  #S\_DEVICE\_INFO,  operationType { profileDownload },  matchingIdSource none : NULL  } |
| CTX\_PARAMS1\_DEVICE\_INFO\_NAI | ctxParamsForCommonAuthentication : {  matchingId #MATCHING\_ID\_EMPTY,  deviceInfo #S\_DEVICE\_INFO\_NAI,  operationType { profileDownload }  } |
| CTX\_PARAMS1\_EVENT\_ID  (CtxParams1) | ctxParamsForCommonAuthentication : {  matchingId #EVENT\_ID\_1,  #S\_DEVICE\_INFO,  matchingIdSource smdsOid : { #S\_SM\_DS\_OID }  } |
| CTX\_PARAMS1\_EVENT\_ID\_IMEI  (CtxParams1) | ctxParamsForCommonAuthentication : {  matchingId #EVENT\_ID\_1,  #S\_DEVICE\_INFO\_IMEI,  matchingIdSource smdsOid : { #S\_SM\_DS\_OID }  } |
| CTX\_PARAMS1\_IMEI  (CtxParams1) | ctxParamsForCommonAuthentication : {  #S\_DEVICE\_INFO\_IMEI,  matchingIdSource none : NULL  } |
| CTX\_PARAMS1\_MATCH\_ID  (CtxParams1) | ctxParamsForCommonAuthentication : {  matchingId #MATCHING\_ID\_1,  #S\_DEVICE\_INFO,  matchingIdSource activationCode : NULL  } |
| CTX\_PARAMS1\_RPM  (CtxParams1) | ctxParamsForCommonAuthentication : {  #S\_DEVICE\_INFO,  operationType { rpm },  matchingIdSource none : NULL  } |
| CTX\_PARAMS1\_RPM\_ICCID1 | ctxParamsForCommonAuthentication : {  #S\_DEVICE\_INFO,  operationType {rpm},  iccid #ICCID\_OP\_PROF1  matchingIdSource none : NULL  } |
| DPI\_RPM1 | /dpi\_1 |
| EF\_UST1 | 0x0A 2E 14 8C E7 32 04 00 00 00 00 00 00  NOTE: Service n°17 (GID1) and n°18 (GID2) not available. |
| EF\_UST2 | 0x0A 2E 17 8C E7 32 04 00 00 00 00 00 00  NOTE: Service n°17 (GID1) and n°18 (GID2) available. |
| EID1 | 0x89 04 90 32 12 34 51 23 45 12 34 56 78 90 12 35 |
| ENTERPRISE\_NAME1 | Enterprise Name 1 |
| ESIM\_PORT\_INIT | eSIM Port 0 to be used for subsequent eSIM initialization APDUs after MANAGE LSI (Configure LSI) command |
| ESIM\_PORT\_INIT\_1 | eSIM Port 1 to be used for subsequent eSIM initialization APDUs after MANAGE LSI (Configure LSI) command |
| EUICC\_SIGNED1 | {  transactionId <S\_TRANSACTION\_ID>,  serverAddress #TEST\_DP\_ADDRESS1,  serverChallenge <S\_SMDP\_CHALLENGE>,  euiccInfo2 #R\_EUICC\_INFO2, -- check only that the field is present and has a valid TLV asn.1 structure  ctxParams1 #CTX\_PARAMS1  } |
| EVENT\_ID\_1 | 07399-BGH7E-T8779 |
| GID1 | 0x47 53 4D 41 |
| GID2 | 0x52 53 50 FF |
| HOST\_ID | 0x47 53 4D 41 20 53 4D 2D 58 58  NOTE: 'GSMA SM-XX' in ASCII. |
| ICCID\_OP\_PROF1 | 0x98 92 09 01 21 43 65 87 09 F5 |
| ICCID\_OP\_PROF2 | 0x98 92 09 01 32 54 76 98 10 F9 |
| ICCID\_OP\_PROF3 | 0x98 92 09 01 43 65 87 09 21 F5 |
| ICCID\_OP\_PROF4 | 0x98 92 09 01 54 76 98 10 32 F9 |
| ICCID\_OP\_PROF9 | 0x98 92 09 01 21 43 65 87 76 F5 |
| ICCID\_OP\_PROF10 | 0x98 10 99 09 00 21 43 65 87 79 |
| ICCID\_OP\_PROFX | 0x98 92 09 01 43 65 87 09 FF FF |
| ICCID\_UNKNOWN | 0x98 92 01 0A 21 43 65 87 09 F8 |
| ICON\_JPG | ICON\_JPG.jpg as defined in Annex H |
| ICON\_OP\_PROF1 | profile\_O1.png as defined in Annex H |
| ICON\_OP\_PROF2 | profile\_O2.png as defined in Annex H |
| ICON\_OP\_PROF3 | profile\_O3.png as defined in Annex H |
| ICON\_OP\_PROF4 | profile\_O4.png as defined in Annex H |
| ICON\_OP\_PROF5 | profile\_O5.png as defined in Annex H |
| IMSI\_OP\_PROF1 | 0x08 29 99 18 11 32 54 76 98 |
| IMSI\_OP\_PROF2 | 0x08 29 99 28 11 32 54 76 97 |
| IMSI\_OP\_PROF3 | 0x08 29 99 28 11 32 54 76 96 |
| IMSI\_OP\_PROF4 | 0x08 29 99 48 43 65 87 09 21 |
| IMSI\_OP\_PROF9 | 0x08 29 99 98 43 65 87 09 21 |
| INVALID\_KEY\_TYPE | 0x80 |
| INVALID\_REMOTE\_OP\_ID | 8 |
| ISD\_R\_AID | 0xA0 00 00 05 59 10 10 FF FF FF FF 89 00 00 01 00 |
| KEY\_LENGTH | 0x10 |
| KEY\_TYPE | 0x88 |
| MATCHING\_ID\_1 | 04386-AGYFT-A74Y8-3F815 |
| MATCHING\_ID\_2 | 04386-AGYFT-A74Y8-3F816 |
| MATCHING\_ID\_EMPTY | <zero-length string> |
| MCC\_MNC\_WILDCARD | 0x92 F9 EE |
| MCC\_MNC1 | 0x92 F9 18 |
| MCC\_MNC2 | 0x92 F9 28 |
| MCC\_MNC3 | 0x92 F9 28 |
| MCC\_MNC4 | 0x92 F9 48 |
| MCC\_MNC9 | 0x92 F9 98 |
| MEP\_REFRESH\_FLAG | TRUE or FALSE depending on the TC capability. |
| MNO\_SCP80\_AUTH\_KEY | 0x11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 10 |
| MNO\_SCP80\_DATA\_ENC\_KEY | 0x99 AA BB CC DD EE FF 10 11 22 33 44 55 66 77 88 |
| MNO\_SCP80\_ENC\_KEY | 0x66 77 88 99 AA BB CC DD 11 22 33 44 55 EE FF 10 |
| NAME\_OP\_PROF1 | Operational Profile Name 1 |
| NAME\_OP\_PROF2 | Operational Profile Name 2 |
| NAME\_OP\_PROF3 | Operational Profile Name 3 |
| NAME\_OP\_PROF4 | Operational Profile Name 4 |
| NAME\_OP\_PROF5 | Operational Profile Name 5 |
| NAME\_OP\_PROF9 | Operational Profile Name 9 |
| NICKNAME1 | Nickname 1 |
| NICKNAME2 | Nickname 2 |
| NICKNAME3 | Nickname 3 |
| OWNER\_OP\_PROF1 | { mccMnc #MCC\_MNC1 } |
| OWNER\_OP\_PROF2 | { mccMnc #MCC\_MNC2 } |
| PO1\_PIN1 | 0x32 34 36 38 FF FF FF FF |
| PPK\_ENC\_INV\_SIZE | 0x01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 0D 0E 0F 10 0D 0E 0F 10 |
| PPK\_INIT\_MAC\_INV\_SIZE | 0x05 0A 04 0B 03 0C 02 0D 01 0E 00 0F 09 01 08 02 09 01 08 02 09 01 08 02 |
| PPK\_MAC\_INV\_SIZE | 0x01 0E 00 0F 09 01 08 02 05 0A 04 0B 03 0C 02 0D 03 0C 02 0D 03 0C 02 0D |
| REMOTE\_OP\_ID\_INSTALL | 1 |
| RSP\_VERSION\_LOWEST\_H | This field is set to the lowest RSP version as defined in SGP.22 Annex O, encoded as the value part of an ASN.1 VersionType (i.e. 0x02 01 00) |
| S\_DEVICE\_INFO | deviceInfo {  tac #S\_TAC,  deviceCapabilities {  gsmSupportedRelease '050000'H,  utranSupportedRelease '080000'H,  cdma2000onexSupportedRelease '010000'H,  cdma2000hrpdSupportedRelease '010000'H,  cdma2000ehrpdSupportedRelease '020000'H,  eutranSupportedRelease '020000'H,  contactlessSupportedRelease '090000'H,  nrEpcSupportedRelease '170000'H,  nr5gcSupportedRelease '170000'H,  eutran5gcSupportedRelease '170000'H,  lpaSvn '030000'H,  catSupportedClasses '111111111111111111111111111111'B,  euiccFormFactorType #IUT\_EUICC\_FF\_TYPE  },  lpaRspCapability '11’B  } |
| S\_DEVICE\_INFO\_IMEI | deviceInfo {  tac #S\_TAC,  deviceCapabilities {  gsmSupportedRelease '050000'H,  utranSupportedRelease '080000'H,  cdma2000onexSupportedRelease '01000'H,  eutranSupportedRelease '020000'H,  lpaSvn '030000'H,  catSupportedClasses '111111111111111111111111111111'B,  euiccFormFactorType #IUT\_EUICC\_FF\_TYPE  },  imei #S\_IMEI,  lpaRspCapability '11’B  } |
| S\_DEVICE\_INFO\_NAI | deviceInfo DeviceInfo {  tac #S\_TAC,  deviceCapabilities {  gsmSupportedRelease '050000'H,  utranSupportedRelease '080000'H,  cdma2000onexSupportedRelease '010000'H,  cdma2000hrpdSupportedRelease '010000'H,  cdma2000ehrpdSupportedRelease '020000'H,  eutranEpcSupportedRelease '020000'H, contactlessSupportedRelease '090000'H,  rspCrlSupportedVersion, #RSP\_SVN\_H  nrEpcSupportedRelease '0F0000'H,  nr5gcSupportedRelease '100000'H,  eutran5gcSupportedRelease '100000'H,  -- No lpaSvn field  -- No catSupportedClasses field  -- No euiccFormFactorType field  deviceAdditionalFeatureSupport {  naiSupport '100000'H  }  }  } |
| S\_ENTERPRISE\_OID | 2.888.99 |
| S\_ENTERPRISE\_OID2 | 2.888.999 |
| S\_EUICC\_CHALLENGE | 0x01 02 03 04 05 06 07 08 01 02 03 04 05 06 07 08 |
| S\_IMEI | 0x00 00 00 00 11 11 11 11 |
| S\_PROFILE\_OWNER\_OID | 2.888.9 |
| S\_PROFILE\_OWNER\_OID2 | 2.888.99 |
| S\_PROFILE\_OWNER\_OIDX | 2.888.999 |
| S\_SERVER\_SVN\_H | 0x03 00 00 |
| S\_SM\_DP+\_OID | 2.999.10 |
| S\_SM\_DP+\_OID2 | 2.999.12 |
| S\_SM\_DS\_OID | 2.999.15 |
| S\_TAC | 0x00 00 00 00 |
| SIMA\_RESULT\_OK | simaresp EUICCResponse ::= {  peStatus {  {status ok}  }  } |
| SP\_NAME1 | SP Name 1 |
| SP\_NAME2 | SP Name 2 |
| SP\_NAME3 | SP Name 3 |
| SP\_NAME4 | SP Name 4 |
| SP\_NAME9 | SP Name 9 |
| SSD\_AID | 0xA0 00 00 05 59 10 10 01 02 73 64 56 61 6C 75 65 |
| TARGET\_ESIM\_PORT | Target eSIM Port to be used for ES10c.EnableProfileRequest and ES10c.DisableProfileRequest |
| TEST\_DP\_ADDRESS1 | testsmdpplus1.example.com |
| TEST\_DP\_ADDRESS2 | testsmdpplus2.example.com |
| TEST\_DP\_ADDRESS3 | testsmdpplus3.example.com |
| TEST\_DP\_ADDRESS4 | testsmdpplus4.example.com |
| TEST\_HRI\_ADDRESS1 | testhriserver1.example.com |
| TEST\_HRI\_ADDRESS3 | testhriserver3.example.com |
| TEST\_PCMP\_ADDRESS1 | testpcmp1.example.com |
| TEST\_PCMP\_ADDRESS3 | testpcmp3.example.com |
| TEST\_ROOT\_DS\_ADDRESS | testrootsmds.example.com |
| UNKNOWN\_BPP\_SEGMENT | 0xC9 05 01 02 03 04 05 |
| UPP\_OP\_PROF1 | The Unprotected Profile Package related to the PROFILE\_OPERATIONAL1 (see Annex E). |
| UPP\_OP\_PROF2 | The Unprotected Profile Package related to the PROFILE\_OPERATIONAL2 (see Annex E). |
| UPP\_OP\_PROF3 | The Unprotected Profile Package related to the PROFILE\_OPERATIONAL3 (see Annex E). |
| UPP\_OP\_PROF4 | The Unprotected Profile Package related to the PROFILE\_OPERATIONAL4 (see Annex E). |
| UPP\_OP\_PROF9 | The Unprotected Profile Package related to the PROFILE\_OPERATIONAL9 (see Annex E). |
| UPP\_OP\_PROF10 | The Unprotected Profile Package related to the PROFILE\_OPERATIONAL10 (see Annex E). |
| USIM\_AID | 0xA0 00 00 00 87 10 02 FF 33 FF 01 89 00 00 01 00 |
| VENDOR\_SPECIFIC\_EXTENSION1 | VendorSpecificExtension : {  {  vendorOid 2.999.16,  vendorSpecificData ‘C1020304’  }  } |
| VENDOR\_SPECIFIC\_EXTENSION2 | VendorSpecificExtension : {  {  vendorOid 2.999.17,  vendorSpecificData ‘02020202’  }  } |

A.2 Test Certificates and Test Keys

All ECC certificates and keys described below are based on descriptions and security requirements (such as algorithms to be used) from SGP.22[2].

NOTE: SGP.26 [25] contains test keys, valid test certificates and instructions for how to generate invalid certificates. All test keys and test certificates used in the present document are bundled with SGP.26 [25].

| Name | Description |
| --- | --- |
| CERT\_CI\_SIG | Certificate of the CI for its Public ECDSA Key. |
| CERT\_CI\_SubCA\_SIG | The intermediate CI SubCA Certificate through which #CERT\_EUM\_SIG, #CERT\_S\_SM\_DP\_SubCA\_SIG or #CERT\_S\_SM\_DPauth\_SIG and the #CERT\_S\_SM\_DPpb\_SIG are chained to the #CERT\_CI\_SIG in Variants B and C. This certificate contains the same Extension for subjectAltName value as in #CERT\_CI\_SIG of that same GSMA CI. |
| CERT\_EUICC\_SIG | Certificate of the eUICC for its Public ECDSA key  CERT.EUICC.SIG in the X.509 format signed by the EUM with SK.EUM.SIG or SK.EUMSubCA.SIG. |
| CERT\_EUM\_SIG | Certificate of the EUM for its Public ECDSA key  CERT.EUM.SIG in the X.509 format signed by the requested CI with SK.CI.SIG or SK.CISubCA.SIG. |
| CERT\_EUM\_SubCA\_SIG | The intermediate SubCA Certificate through which #CERT\_EUICC\_SIG is chained to the #CERT\_EUM\_SIG in Variants A and C. This certificate contains the same Extension for subjectAltName value as in #CERT\_EUM\_SIG. |
| CERT\_S\_SM\_DP\_SubCA\_SIG | The intermediate SubCA Certificate through which #CERT\_S\_SM\_DPauth\_SIG and the #CERT\_S\_SM\_DPpb\_SIG are chained to the #CERT\_CI\_SIG in Variant A or to the #CERT\_CI**\_**SubCA\_SIG in Variant C. This certificate contains the Extension for subjectAltName as OID #S\_SM\_DP+\_OID. |
| CERT\_S\_SM\_DP\_SubCAList\_SIG  (CertificateChain) | #CERT\_S\_SM\_DP\_SubCA\_SIG, #CERT\_CI\_SubCA\_SIG |
| CERT\_S\_SM\_DP2auth\_SIG | Certificate of the S\_SM-DP+ for its Public ECDSA key used for SM‑DP+ authentication. This certificate contains the OID #S\_SM\_DP+\_OID2. |
| CERT\_S\_SM\_DP2pb\_SIG | Certificate of the S\_SM-DP+ for its Public ECDSA key used for Profile Package Binding. This certificate contains the OID #S\_SM\_DP+\_OID2. |
| CERT\_S\_SM\_DPauth\_SIG | Certificate of the S\_SM-DP+ for its Public ECDSA key used for SM‑DP+ authentication. This certificate contains the OID #S\_SM\_DP+\_OID. |
| CERT\_S\_SM\_DPauth\_INV\_CURVE | Certificate of the S\_SM-DP+ for its Public ECDSA key used for Authentication. This certificate contains the OID #S\_SM\_DP+\_OID and a public key based on a curve different from the allowed curves defined in SGP.22[2], section 2.6. |
| CERT\_S\_SM\_DPauth\_INV\_SIGN | Invalid certificate of the S\_SM-DP+ for its Public ECDSA key used for authentication. This certificate contains the OID #S\_SM\_DP+\_OID and contains an invalid signature (i.e. not generated with the #SK\_CI\_SIG but with the same tag and length as a valid signature). |
| CERT\_S\_SM\_DPpb\_SIG | Certificate of the S\_SM-DP+ for its Public ECDSA key used for Profile Package Binding. This certificate contains the OID #S\_SM\_DP+\_OID. |
| CERT\_S\_SM\_DPpb\_INV\_CURVE | Certificate of the S\_SM-DP+ for its Public ECDSA key used for Profile Package Binding. This certificate contains the OID #S\_SM\_DP+\_OID and a public key based on a curve different from the allowed curves defined in SGP.22[2], section 2.6. |
| CERT\_S\_SM\_DPpb\_INV\_SIGN | Invalid certificate of the S\_SM-DP+ for its Public ECDSA key used for Profile Package Binding. This certificate contains the OID #S\_SM\_DP+\_OID and contains an invalid signature (i.e. not generated with the #SK\_CI\_SIG but with the same tag and length as a valid signature). |
| CERT\_S\_SM\_DSauth\_SIG | Certificate of the S\_SM-DS for its Public ECDSA key used for SM‑DS authentication. This certificate contains the OID #S\_SM\_DS\_OID. |
| CERT\_S\_SM\_DSauth\_INV\_CURVE | Certificate of the S\_SM-DS for its Public ECDSA key used for Authentication. This certificate contains the OID #S\_SM\_DS\_OID and a public key based on a curve different from the allowed curves defined in SGP.22[2], section 2.6. |
| CERT\_S\_SM\_DSauth\_INV\_SIGN | Invalid certificate of the S\_SM-DS for its Public ECDSA key used for SM‑DS authentication. This certificate contains an invalid signature, (i.e. not generated with the #SK\_CI\_SIG but with the same tag and length as a valid signature). |
| CRL\_LIST | The list of CRLs needed to verify the revocation status of each Certificate that contains a cRLDistributionPoints extension in the returned Certificate chain. |
| PK\_CI\_SIG | Public Key of the CI, contained within #CERT\_CI\_SIG. |
| PK\_EUICC\_SIG | Public Key of the eUICC, contained within #CERT\_EUICC\_SIG. |
| SK\_CI\_SIG | Private Key of the CI. |
| SK\_EUICC\_SIG | Private key of the eUICC for creating signatures. |
| SK\_S\_SM\_DPauth\_SIG | Private Key of the S\_SM-DP+ for creating signatures for SM-DP+ authentication. |
| SK\_S\_SM\_DPpb\_SIG | Private key of the S\_SM-DP+ used to provide signatures for Profile binding. |
| SK\_S\_SM\_DSauth\_SIG | Private Key of the S\_SM-DS for creating signatures for SM-DS authentication. |

Annex B Dynamic Content

| **Variable** | **Description** |
| --- | --- |
| ANY\_PORT\_VALUE | Any valid positive INTEGER value |
| ANY\_SW\_IN\_ERROR | Any Status Word in error (different from 0x9000). |
| ATR | ATR received from eUICC after RESET |
| BPP | Content of a Bound Profile Package to download within the eUICC. |
| BPP\_SEG\_A0 | Bound Profile Package TLV segment containing the tag and length fields of the firstSequenceOf87 TLV plus the first 0x87 TLV containing the ConfigureISDP command. |
| BPP\_SEG\_A1 | Bound Profile Package following TLV segment array, as defined in SGP.22 [2] – section 2.5.5:   * array first element containing the Tag and length fields of the sequenceOf88 TLV * array following elements containing each of the ‘88’ TLVs containing the StoreMetadata command |
| BPP\_SEG\_A2 | Bound Profile Package TLV segment containing the Tag and length fields of the secondSequenceOf87 TLV plus the first '87' TLV, containing the ReplaceSessionKeys command. |
| BPP\_SEG\_A3 | Bound Profile Package following TLV segment array, as defined in SGP.22 [2] – section 2.5.5:   * array first element containing the tag and length fields of the sequenceOf86 TLV * array following elements containing each of the '86' TLVs containing the Protected Profile Package (PPP) |
| BPP\_SEG\_INIT | Bound Profile Package TLV segment containing the tag and length fields of the BoundProfilePackage TLV plus the initialiseSecureChannelRequest command. |
| C\_APDUS\_SCRIPT | List of Command APDUs formatted as an expanded structure with definite length coding as defined in ETSI TS 102 226 [14]. |
| CC | SCP80 cryptographic checksum as defined in ETSI TS 102 225 [13] (8 bytes long). |
| CHANNEL\_NUMBER | The logical channel number newly opened in the eUICC. If no logical channel is opened, the value is set to 0x00 (i.e. Basic Channel). There is one value per LSI. |
| CI\_PKI\_RANDOM | Random Subject Key Identifier of the PK CI different from all the PK CI Identifiers defined in SGP.26 [25]. This random value has the same length as the ones defined in SGP.26 [25]. |
| EUICC\_CHALLENGE | Random eUICC challenge, coded as asn.1 OCTET STRING, 16 bytes. |
| EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING | List of CI Public Key Identifiers supported on the eUICC for signature creation that can be verified by a certificate chain Variant O, coded as ASN.1 sequence of SubjectKeyIdentifier. The CI Public Key Identifiers are from the list of possible identifiers as defined in SGP.26 [25]. This list can be empty. |
| EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3 | List of CI Public Key Identifiers supported on the eUICC for signature creation that can be verified by a certificate chain Variant A, B or C, coded as ASN.1 sequence of SubjectKeyIdentifier. The CI Public Key Identifiers are from the list of possible identifiers as defined in SGP.26 [25]. |
| EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION | List of CI Public Key Identifiers supported on the eUICC for signature verification, coded as ASN.1 sequence of SubjectKeyIdentifier. The CI Public Key Identifiers are from the list of possible identifiers as defined in SGP.26 [25]. |
| EUICC\_CI\_PK\_ID\_TO\_BE\_USED | CI Public Key Identifier to be used by the eUICC for signature, coded as ASN.1 sequence of SubjectKeyIdentifier, selected from <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING>. |
| EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3 | CI Public Key Identifier to be used by the eUICC for signature, coded as ASN.1 sequence of SubjectKeyIdentifier, selected from <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3>. |
| EUICC\_CS\_SIGNATURE | The eUICC cancel session signature computed using the #SK\_EUICC\_SIG across the EuiccCancelSessionSigned present in the CancelSessionResponse structure |
| EUICC\_RSP\_CAPABILITY | EuiccRspCapability, coded as ASN.1 BIT STRING to be used for indication of additionalProfile, loadcrlSupport, rpmSupport , testProfileSupport, etc. |
| EUICC\_SIGN\_PIR | The eUICC signature of the Profile Installation Result (PIR). The input data used to generate the <EUICC\_SIGN\_PIR> is the profileInstallationResultData TLV. |
| EUICC\_SIGN\_RPR | The eUICC signature of the Load RPM Package Result (RPR). The input data used to generate the <EUICC\_SIGN\_RPR> is the loadRpmPackageResultDataSigned and smdpSignature3.  euiccSignRPR shall be created using the SK.EUICC.SIG and verified using the PK.EUICC.SIG. |
| EUICC\_SIGNATURE1 | The eUICC signature 1 (euiccSignature1) computed using #SK\_EUICC\_SIG across the euiccSigned1 present in the AuthenticateServerResponse structure, coded as ASN.1 OCTET STRING. |
| EUICC\_SIGNATURE2 | The eUICC signature 2 (euiccSignature2) computed using the #SK\_EUICC\_SIG across the following data objects:   * euiccSigned2 * smdpSignature2 present in the PrepareDownloadRequest structure |
| EXT\_CARD\_RESOURCE | Extended Card Resource Information according to ETSI TS 102 226 [14], coded as ASN.1 OCTET STRING. 'Number of installed application' value field is '00'. |
| FREE\_MEM\_OP\_PROF\_INSTALLED | Non-volatile memory (tag 0x82) available in the eUICC when two or more PROFILE\_OPERATIONAL are installed. |
| FREE\_MEM\_OP\_PROF1\_DELETED | Non-volatile memory (tag 0x82) available in the eUICC after PROFILE\_OPERATIONAL1 deletion. |
| FREE\_MEM\_OP\_PROF1\_INSTALLED | Non-volatile memory (tag 0x82) available in the eUICC when only PROFILE\_OPERATIONAL1 is installed. |
| FREE\_MEMORY\_NO\_PROFILE | Non-volatile memory (tag 0x82) available in the eUICC when there is no Profile installed. |
| INVALID\_TRANSACTION\_ID | A Transaction Identifier generated by the S\_SM-DP+ or the S\_SM‑DS that SHALL be different from <S\_TRANSACTION\_ID> if exists. Otherwise, a random value is generated. |
| ISD\_P\_AID | The ISD-P AID newly created in the eUICC. This AID value is in the range from 0xA0 00 00 05 59 10 10 FF FF FF FF 89 00 00 10 00 to 0xA0 00 00 05 59 10 10 FF FF FF FF 89 00 FF FF 00. Last byte is set to '00' as defined in SGP.02[1]. |
| ISD\_P\_AID1 | The ISD-P AID created in the eUICC for the PROFILE\_OPERATIONAL1. This AID value belongs to the range from 0xA0 00 00 05 59 10 10 FF FF FF FF 89 00 00 10 00 to 0xA0 00 00 05 59 10 10 FF FF FF FF 89 00 FF FF 00. Last byte is set to '00' as defined in SGP.02[1]. |
| ISD\_P\_AID2 | The ISD-P AID created in the eUICC for the PROFILE\_OPERATIONAL2. This AID value belongs to the range from 0xA0 00 00 05 59 10 10 FF FF FF FF 89 00 00 10 00 to 0xA0 00 00 05 59 10 10 FF FF FF FF 89 00 FF FF 00. Last byte is set to '00' as defined in SGP.02[1]. |
| ISD\_P\_AID3 | The ISD-P AID created in the eUICC for the PROFILE\_OPERATIONAL3. This AID value belongs to the range from 0xA0 00 00 05 59 10 10 FF FF FF FF 89 00 00 10 00 to 0xA0 00 00 05 59 10 10 FF FF FF FF 89 00 FF FF 00. Last byte is set to '00' as defined in SGP.02[1]. |
| ISD\_P\_AID4 | The ISD-P AID created in the eUICC for the PROFILE\_OPERATIONAL4. This AID value belongs to the range from 0xA0 00 00 05 59 10 10 FF FF FF FF 89 00 00 10 00 to 0xA0 00 00 05 59 10 10 FF FF FF FF 89 00 FF FF 00. Last byte is set to '00' as defined in SGP.02[1]. |
| ISD\_P\_AIDX | An invalid ISD-P AID not present on the eUICC. This AID value is in the range from 0xA0 00 00 05 59 10 10 FF FF FF FF 89 00 00 10 00 to 0xA0 00 00 05 59 10 10 FF FF FF FF 89 00 FF FF 00. |
| L | Exact length of the corresponding tag or of the remaining data. |
| LSI\_COMMAND\_ACTION | Action returned in the proactive command LSI COMMAND |
| LSI\_NUMBER | LSI numbers returned in the proactive command LSI COMMAND |
| LSI\_SUPPORT | TRUE if ‘LSI support’ is present in the ATR |
| MEP\_LSI\_OPTIONS | LSI Options returned in the MANAGE LSI(Configure LSI) response |
| MEP\_MAX\_LSIS | Maximum number of LSIs supported for Enabled Profiles |
| MEP\_MODE | The ‘Jointly supported MEP mode’ returned in the MANAGE LSI(Configure LSI) response |
| MEP-A2\_TARGET\_ESIM\_PORT | Target eSIM Port returned in the ES10c.EnableProfileResponse for MEP-A2. |
| MNO\_SCP80\_COUNTER | SCP80 counter of the MNO-SD related to the KVN 0x01 (5 bytes long). Initial value is set to 0x00 00 00 00 01 and is incremented by one each time a secured packet is sent. |
| NB\_EXECUTED\_C\_APDUS | Number of executed Command TLV objects as defined in ETSI TS 102 226 [14]. |
| NEXT\_EUICC\_CERT\_IN\_CHAIN | The Certificate certifying the eUICC Certificate #CERT\_EUICC\_SIG. |
| NOTIF\_SEQ\_NO\_DE1 | The Sequence Number of the Delete Notification related to the PROFILE\_OPERATIONAL1. |
| NOTIF\_SEQ\_NO\_DE2 | The Sequence Number of the Delete Notification related to the PROFILE\_OPERATIONAL2. |
| NOTIF\_SEQ\_NO\_DI1 | The Sequence Number of the Disable Notification related to the PROFILE\_OPERATIONAL1. |
| NOTIF\_SEQ\_NO\_EN1 | The Sequence Number of the Enable Notification related to the PROFILE\_OPERATIONAL1. |
| NOTIF\_SEQ\_NO\_EN1\_RPM | The Sequence Number of the RPM Enable Notification related to the PROFILE\_OPERATIONAL1. |
| NOTIF\_SEQ\_NO\_EN2 | The Sequence Number of the Enable Notification related to the PROFILE\_OPERATIONAL2. |
| NOTIF\_SEQ\_NO\_IN1 | The Sequence Number of the Install Notification related to the PROFILE\_OPERATIONAL1. |
| NOTIF\_SEQ\_NO\_IN1\_PIR | The Sequence Number of the Install Notification (PIR) related to the PROFILE\_OPERATIONAL1. |
| NOTIF\_SEQ\_NO\_IN2 | The Sequence Number of the Install Notification related to the PROFILE\_OPERATIONAL2. |
| NOTIF\_SEQ\_NO\_IN2\_PIR | The Sequence Number of the Install Notification (PIR) related to the PROFILE\_OPERATIONAL2. |
| NOTIF\_SEQ\_NO\_PROF1\_RPR | The Sequence Number of the Notification RPM Package Result related to the PROFILE\_OPERATIONAL1. |
| NOTIF\_SEQ\_NO2\_DE1 | The Sequence Number of the second Delete Notification related to the PROFILE\_OPERATIONAL1. |
| NOTIF\_SEQ\_NO2\_DI1 | The Sequence Number of the second Disable Notification related to the PROFILE\_OPERATIONAL1. |
| NOTIF\_SEQ\_NO2\_EN1 | The Sequence Number of the second Enable Notification related to the PROFILE\_OPERATIONAL1. |
| OT\_SK\_S\_SM\_DP+\_ECKA | One-time Private Key generated by the S\_SM-DP+ for ECKA. Depending on the eUICC configuration, this key is based on NIST P-256, brainpoolP256r1 or FRP256V1. |
| OTHER\_EUICC\_CERT\_IN\_CHAIN | Remaining part of the certificate chain certifying the <NEXT\_EUICC\_CERT\_IN\_CHAIN>. |
| OTPK\_S\_SM\_DP+\_ECKA | One-time Public Key generated by the S\_SM-DP+ for ECKA. Depending on the eUICC configuration, this key is based on NIST P-256, brainpoolP256r1 or FRP256V1. |
| PPK\_ENC | Random PPK-ENC value (16 bytes key length). This value is different from <S\_ENC> value. |
| PPK\_INIT\_MAC | Random initial MAC chaining value (16 bytes). This value is different from the <S\_MAC\_CHAIN> value. |
| PPK\_MAC | Random PPK-MAC value (16 bytes key length). This value is different from <S\_MAC> value. |
| PPR\_IDS | Forbidden Profile Policy Rules. This PPR list MAY be empty or MAY contain either PPR1 or PPR2 or both. |
| PROPRIETARY\_DATA | Proprietary Data returned by the eUICC as part of FCI template. |
| RANDOM\_SM\_DP+\_SIGN | Random SM-DP+ signature (i.e. content of the tag 0x5F37) with a size corresponding to a valid one. |
| RANDOM\_SM\_DS\_SIGN | Random SM-DS signature (i.e. content of the tag 0x5F37) with a size corresponding to a valid one. |
| S\_ENC | SCP03T Encryption Session key (128 bits length) resulting from the key agreement with eUICC. |
| S\_HASHED\_CC | Hashed Confirmation Code generated by the LPA. |
| S\_INIT\_MAC | SCP03T Initial MAC chaining value (128 bits length) resulting from the key agreement with eUICC. |
| S\_MAC | SCP03T MACing Session key (128 bits length) resulting from the key agreement with eUICC. |
| S\_MAC\_CHAIN | Current MAC chaining value used for SCP03t BPP protection. |
| S\_SM\_DP+\_SIGN | The S\_SM-DP+ signature (smdpSign), computed using the #SK\_S\_SM\_DPpb\_SIG across the following data objects:   * remoteOpId * transactionId * controlRefTemplate * smdpOtpk * euiccOtpk, as provided earlier in the prepareDownloadResponse data object |
| S\_SM\_DP+\_SIGNATURE2 | The ASN.1 OCTET STRING encoded SM-DP+ signature 2 (field smdpSignature2) computed using the private key related to the server certificate (field smdpCertificate) present in the PrepareDownloadRequest structure. This signature SHALL be generated across the following data objects:   * smdpSignature2 * euiccSignature1 present in the AuthenticateServerResponse structure |
| S\_SM\_DP+\_SIGNATURE3 | The ASN.1 OCTET STRING encoded SM-DP+ signature 3 (field smdpSignature3) is computed using the SK.DPauth.SIG. This signature SHALL be generated across the following data objects:   * smdpSigned3 * euiccSignature1 present in the AuthenticateServerResponse structure |
| S\_SMDP\_CHALLENGE | The SM-DP+ Challenge (serverChallenge) randomly chosen by the simulated SM-DP+ to be signed later by the eUICC for the eUICC authentication, coded as ASN.1 OCTET STRING of 16 bytes. |
| S\_SMDP\_SIGNATURE1 | The ASN.1 OCTET STRING encoded SM-DP+ signature (field serverSignature1) computed using the private key related to the server certificate (field serverCertificate) present in the AuthenticateServerRequest structure. |
| S\_SMDS\_CHALLENGE | The SM-DS Challenge (serverChallenge) randomly chosen by the simulated SM-DS to be signed later by the eUICC for the eUICC authentication, coded as ASN.1 OCTET STRING of 16 bytes. |
| S\_SMDS\_SIGNATURE1 | The SM-DS signature 1 (serverSignature1) computed using #SK\_S\_SM\_DSauth\_SIG across the serverSigned1 present in the AuthenticateServerRequest structure, coded as ASN.1 OCTET STRING. |
| S\_TRANSACTION\_ID | The TransactionID (Unique Transaction Identifier) generated by the (S\_)SM-DP+, or (S\_)SM-DS which is used to uniquely identify the RSP session and to correlate the multiple ESXX request messages that belong to the same RSP session. This value (binary value) can start from 0x01 and can be increased by 1 each time a Profile is downloaded in the eUICC. 1-16 bytes (ASN.1 OCTET STRING). |
| SEQ\_NUMBER | Sequence Number related to a Notification Metadata generated by the eUICC. |
| SHS | Shared Secret resulting from the key agreement with eUICC. |
| STORE\_DATA\_BLOCK\_NUM | The STORE DATA block number coded sequentially from 0x00 to 0xFF. If the value 0xFF has been reached and more STORE DATA commands are needed to complete the transfer, the numbering restarts and the next STORE DATA block number is set to 0x00. |
| TBS\_EUICC\_NOTIF\_SIG | The eUICC signature generated over tbsOtherNotification. NotificationMetadata, coded as ASN.1 OCTET STRING. |
| TRE\_PROPERTIES | The value of the treProperties field in EUICCInfo2. |
| TRE\_REFERENCE | The value of the treProductReference field in EUICCInfo2. |

Annex C Methods And Procedures

This section describes methods and procedures used in the interfaces compliance test cases. They are part of test cases and SHALL not be executed in standalone mode.

C.1 Methods

If the method is used in the “expected result” column, all parameters SHALL be verified by the simulated entity (test tool). If the method is used in the “Sequence / Description” column, the command SHALL be generated by the simulated entity.

|  |  |
| --- | --- |
| Method | MTD\_AUTHENTICATE\_SMDP |
| Description | Generate an ASN.1 AuthenticateServerRequest structure according to the input parameters |
| Parameter(s) | * paramServerAddress: the SM-DP+ or the SM-DS FQDN * paramServerChallenge: the SM-DP+ or the SM-DS challenge * paramCtxParams1: the CtxParams1 to use * paramServerSignature: the RSP Server signature * paramServerCertificate: the RSP Server Certificate CERT.XXauth.SIG * paramOtherCertsInChain (OPTIONAL): the remaining part of the CERT.XXauth.SIG certificate chain, if any * paramCrlList: CRL list to use * paramNewCertVariant: true if new certificates variants used |
| Details | Generate the following ASN.1 structure:  req AuthenticateServerRequest ::= {  serverSigned1 {  transactionId <S\_TRANSACTION\_ID>,  euiccChallenge <EUICC\_CHALLENGE>,  serverAddress paramServerAddress,  serverChallenge paramServerChallenge,  sessionContext {  serverSvn #IUT\_RSP\_VERSION\_HIGHEST,  crlStaplingV3Used TRUE,  euiccCiPKIdToBeUsedV3 <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3>  },  serverRspCapability {  crlStaplingV3Support,  cancelForEmptySpnPnSupport  }  },  serverSignature1 paramServerSignature,  euiccCiPKIdToBeUsed  <EUICC\_CI\_PK\_ID\_TO\_BE\_USED>,  serverCertificate paramServerCertificate,  ctxParams1 paramCtxParams1,  otherCertsInChain { paramOtherCertsInChain },  crlList { paramCrlList }  }  otherCertsInChain SHALL be present in the AuthenticateServerRequest only if paramOtherCertsInChain is provided  if paramNewCertVariant is true then euiccCiPKIdToBeUsedV3 SHALL be present in sessionContext and euiccCiPKIdToBeUsed SHALL be omitted. Otherwise, if if paramNewCertVariant is false then euiccCiPKIdToBeUsed SHALL be present and euiccCiPKIdToBeUsedV3 SHALL be omitted in sessionContext. |

|  |  |
| --- | --- |
| Method | MTD\_AUTHENTICATE\_SMDP\_USING\_IUT\_VARIANT |
| Description | Generate an ASN.1 AuthenticateServerRequest structure according to the input parameters, specifying the CI PK to use for signing accordingly with the IUT setting IUT\_EUICC\_CERT\_CHAIN\_VARIANT. |
| Parameter(s) | * paramServerAddress: the SM-DP+ or the SM-DS FQDN * paramServerChallenge: the SM-DP+ or the SM-DS challenge * paramCtxParams1: the CtxParams1 to use * paramServerSignature: the RSP Server signature * paramServerCertificate: the RSP Server Certificate CERT.XXauth.SIG * paramOtherCertsInChain (OPTIONAL): the remaining part of the CERT.XXauth.SIG certificate chain, if any * paramCrlList: CRL list to use |
| Details | Generate the following ASN.1 structure:  req AuthenticateServerRequest ::= {  serverSigned1 {  transactionId <S\_TRANSACTION\_ID>,  euiccChallenge <EUICC\_CHALLENGE>,  serverAddress paramServerAddress,  serverChallenge paramServerChallenge,  sessionContext {  serverSvn #IUT\_RSP\_VERSION\_HIGHEST,  crlStaplingV3Used TRUE,  euiccCiPKIdToBeUsedV3 <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3>  },  serverRspCapability {  crlStaplingV3Support,  cancelForEmptySpnPnSupport  }  },  serverSignature1 paramServerSignature,  euiccCiPKIdToBeUsed  <EUICC\_CI\_PK\_ID\_TO\_BE\_USED>,  serverCertificate paramServerCertificate,  ctxParams1 paramCtxParams1,  otherCertsInChain { paramOtherCertsInChain },  crlList { paramCrlList }  }  otherCertsInChain SHALL be present in the AuthenticateServerRequest only if paramOtherCertsInChain is provided  if IUT\_EUICC\_CERT\_CHAIN\_VARIANT is one of Ov3, A, B or C, then euiccCiPKIdToBeUsedV3 SHALL be present in sessionContext and euiccCiPKIdToBeUsed SHALL be omitted. Otherwise, euiccCiPKIdToBeUsed SHALL be present, and euiccCiPKIdToBeUsedV3 SHALL be omitted in sessionContext. |

|  |  |
| --- | --- |
| Method | MTD\_AUTHENTICATE\_SMDS |
| Description | Generate an ASN.1 AuthenticateServerRequest structure according to the input parameters |
| Parameter(s) | * paramServerAddress: the SM-DP+ or the SM-DS FQDN * paramServerChallenge: the SM-DP+ or the SM-DS challenge * paramCtxParams1: the CtxParams1 to use * paramServerSignature: the RSP Server signature * paramServerCertificate: the RSP Server Certificate CERT.XXauth.SIG * paramOtherCertsInChain (OPTIONAL): the remaining part of the CERT.XXauth.SIG certificate chain, if any * paramCrlList: CRL list to use * paramNewCertVariant: true if new certificates variants used |
| Details | Generate the following ASN.1 structure:  req AuthenticateServerRequest ::= {  serverSigned1 {  transactionId <S\_TRANSACTION\_ID>,  euiccChallenge <EUICC\_CHALLENGE>,  serverAddress paramServerAddress,  serverChallenge paramServerChallenge,  sessionContext {  serverSvn #IUT\_RSP\_VERSION\_HIGHEST,  crlStaplingV3Used TRUE,  euiccCiPKIdToBeUsedV3 <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3>  },  serverRspCapability {  crlStaplingV3Support,  eventListSigningV3Support  }  },  serverSignature1 paramServerSignature,  euiccCiPKIdToBeUsed  <EUICC\_CI\_PK\_ID\_TO\_BE\_USED>,  serverCertificate paramServerCertificate,  ctxParams1 paramCtxParams1,  otherCertsInChain { paramOtherCertsInChain },  crlList { paramCrlList }  }  otherCertsInChain SHALL be present in the AuthenticateServerRequest only if paramOtherCertsInChain is provided  if paramNewCertVariant is true then euiccCiPKIdToBeUsedV3 SHALL be present in sessionContext and euiccCiPKIdToBeUsed SHALL be omitted. Otherwise, if if paramNewCertVariant is false then euiccCiPKIdToBeUsed SHALL be present and euiccCiPKIdToBeUsedV3 SHALL be omitted in sessionContext. |

|  |  |
| --- | --- |
| Method | MTD\_AUTHENTICATE\_SMDS\_USING\_IUT\_VARIANT |
| Description | Generate an ASN.1 AuthenticateServerRequest structure according to the input parameters, specifying the CI PK to use for signing accordingly with the IUT setting IUT\_EUICC\_CERT\_CHAIN\_VARIANT. |
| Parameter(s) | * paramServerAddress: the SM-DP+ or the SM-DS FQDN * paramServerChallenge: the SM-DP+ or the SM-DS challenge * paramCtxParams1: the CtxParams1 to use * paramServerSignature: the RSP Server signature * paramServerCertificate: the RSP Server Certificate CERT.XXauth.SIG * paramOtherCertsInChain (OPTIONAL): the remaining part of the CERT.XXauth.SIG certificate chain, if any * paramCrlList: CRL list to use |
| Details | Generate the following ASN.1 structure:  req AuthenticateServerRequest ::= {  serverSigned1 {  transactionId <S\_TRANSACTION\_ID>,  euiccChallenge <EUICC\_CHALLENGE>,  serverAddress paramServerAddress,  serverChallenge paramServerChallenge,  sessionContext {  serverSvn #IUT\_RSP\_VERSION\_HIGHEST,  crlStaplingV3Used TRUE,  euiccCiPKIdToBeUsedV3 <EUICC\_CI\_PK\_ID\_TO\_BE\_USED\_V3>  },  serverRspCapability {  crlStaplingV3Support,  eventListSigningV3Support  }  },  serverSignature1 paramServerSignature,  euiccCiPKIdToBeUsed  <EUICC\_CI\_PK\_ID\_TO\_BE\_USED>,  serverCertificate paramServerCertificate,  ctxParams1 paramCtxParams1,  otherCertsInChain { paramOtherCertsInChain },  crlList { paramCrlList }  }  otherCertsInChain SHALL be present in the AuthenticateServerRequest only if paramOtherCertsInChain is provided  If IUT\_EUICC\_CERT\_CHAIN\_VARIANT is one of Ov3, A, B or C then euiccCiPKIdToBeUsedV3 SHALL be present in sessionContext and euiccCiPKIdToBeUsed SHALL be omitted. Otherwise, euiccCiPKIdToBeUsed SHALL be present and euiccCiPKIdToBeUsedV3 SHALL be omitted in sessionContext. |

|  |  |
| --- | --- |
| Method | MTD\_CHECK\_AUTH\_ERROR\_RESP |
| Description | Check an ASN.1 AuthenticateServerResponse structure in case of failure according to the input parameters |
| Parameter(s) | * paramAuthenticateErrorCode: the expected error code |
| Details | Verify that the received AuthenticateServerResponse is structured as follow:  resp AuthenticateServerResponse ::= authenticateResponseError : {  transactionId <S\_TRANSACTION\_ID>,  authenticateErrorCode paramAuthenticateErrorCode  }  Verify that the <S\_TRANSACTION\_ID> present in the authenticateResponseError is the same as in the AuthenticateServerRequest. |

|  |  |
| --- | --- |
| Method | MTD\_CHECK\_AUTH\_SERVER\_RESP |
| Description | Check an ASN.1 AuthenticateServerResponse structure according to the input parameters |
| Parameter(s) | * paramServerAddress: the expected SM-DP+ or the SM-DS FQDN * paramServerChallenge: the expected SM-DP+ or SM-DS challenge * paramCtxParams1: the expected CtxParams1 |
| Details | Verify that the received AuthenticateServerResponse is structured as follow:  resp AuthenticateServerResponse ::= authenticateResponseOk : {  euiccSigned1 {  transactionId <S\_TRANSACTION\_ID>,  serverAddress paramServerAddress,  serverChallenge paramServerChallenge,  euiccInfo2 {...},  ctxParams1 paramCtxParams1  },  euiccSignature1 <EUICC\_SIGNATURE1>,  euiccCertificate #CERT\_EUICC\_SIG,  nextCertInChain <NEXT\_EUICC\_CERT\_IN\_CHAIN>,  otherCertsInChain <OTHER\_EUICC\_CERT\_IN\_CHAIN> -- OPTIONAL  }  Verify that euiccInfo2 is present and contains a valid TLV ASN.1 structure. At this level, only the format of the TLV structure is verified.  Verify the <EUICC\_SIGNATURE1> across the euiccSigned1 by using the #PK\_EUICC\_SIG.  Verify that the <S\_TRANSACTION\_ID> present in the euiccSigned1 is the same as in the AuthenticateServerRequest.  If #IUT\_EUICC\_CERT\_CHAIN\_VARIANT = ‘A’ Then   * Verify that <NEXT\_EUICC\_CERT\_IN\_CHAIN> matches #CERT\_EUM\_SubCA\_SIG * Verify that <OTHER\_EUICC\_CERT\_IN\_CHAIN> matches #CERT\_EUM\_SIG   Else If #IUT\_EUICC\_CERT\_CHAIN\_VARIANT = ‘B’ Then   * Verify that <NEXT\_EUICC\_CERT\_IN\_CHAIN> matches #CERT\_EUM\_SIG * Verify that <OTHER\_EUICC\_CERT\_IN\_CHAIN> matches #CERT\_CI\_SubCA\_SIG   Else If #IUT\_EUICC\_CERT\_CHAIN\_VARIANT = ‘C’ Then  *-- meaning that Variant C is supported*   * Verify that <NEXT\_EUICC\_CERT\_IN\_CHAIN> matches #CERT\_EUM\_SubCA\_SIG * Verify that <OTHER\_EUICC\_CERT\_IN\_CHAIN> matches { #CERT\_EUM\_SIG, #CERT\_CI\_SubCA\_SIG }   Else  *-- meaning that Variant O or Variant Ov3 is supported*  • Verify that <NEXT\_EUICC\_CERT\_IN\_CHAIN> matches #CERT\_EUM\_SIG  • Verify that <OTHER\_EUICC\_CERT\_IN\_CHAIN> is not present  End if  Verify that the eUICC Certificate chain leads to the Root CI Key Identifier set in the euiccCiPKIdToBeUsedV3 or in the euiccCiPKIdToBeUsed of the corresponding AuthenticateServerRequest. |

|  |  |
| --- | --- |
| Method | MTD\_CHECK\_SMS\_POR |
| Description | Check the content of the SMS POR containing the response of the ES6.UpdateMetadata request |
| Parameter(s) | * paramExpectedSW: the expected Status Word of the last STORE DATA command |
| Details | Parse and retrieve the SCP80 response packet from the SMS.  SCP80 response status code SHALL be equal to 0x00 – POR OK.  The additional data from the response packet SHALL be formatted as an expanded structure with definite length as defined in ETSI TS 102 226 [14] and contains the following TLV:  AB <L>  80 <L> <NB\_EXECUTED\_C\_APDUS> *-- Number of executed C-APDUs*  23 <L> 00 90 00 *–- R-APDU of the INSTALL FOR PERSONALIZATION command*  23 <L> paramExpectedSW *–- SW of the last STORE DATA command executed*  <NB\_EXECUTED\_C\_APDUS> SHALL be equal to the number of executed C-APDUs (i.e. one INSTALL FOR PERSONALIZATION + n STORE DATA command(s)) |

|  |  |
| --- | --- |
| Method | MTD\_DELETE\_PROFILE |
| Description | Generate the ASN.1 DeleteProfileRequest structure according to the input parameters. |
| Parameter(s) | * paramIccidValue: The ICCID of the Profile to Delete (optional) * paramIsdpAidValue: The ISD-P AID of the Profile to Delete (optional)   Either paramIccidValue or paramIsdpAidValue is passed as a parameter. |
| Details | IF paramIccidValue is provided Then  req DeleteProfileRequest ::= iccid : paramIccidValue  Else  req DeleteProfileRequest ::= isdpAid : paramIsdpAidValue  End if |

|  |  |
| --- | --- |
| Method | MTD\_DISABLE\_PROFILE |
| Description | Generate the ASN.1 DisableProfileRequest structure according to the input parameters. |
| Parameter(s) |  paramIccidValue: The ICCID of the Profile to Disable (optional)   paramIsdpAidValue: The ISD-P AID of the Profile to Disable (optional)   paramRefreshFlag: Boolean, TRUE if refreshFlagSHALL be set, FALSE otherwise  Either paramIccidValue or paramIsdpAidValue is passed as a parameter. |
| Details | IF paramIccidValue is provided Then  req DisableProfileRequest::= {  profileIdentifier iccid : paramIccidValue,  refreshFlag paramRefreshFlag  }  Else  req DisableProfileRequest::= {  profileIdentifier isdpAid : paramIsdpAidValue,  refreshFlag paramRefreshFlag  }  End if |

|  |  |
| --- | --- |
| Method | MTD\_ENABLE\_PROFILE |
| Description | Generate the ASN.1 EnableProfileRequest structure according to the input parameters. |
| Parameter(s) | * paramIccidValue: The ICCID of the Profile to Enable (optional) * paramIsdpAidValue: The ISD-P AID of the Profile to Enable (optional) * paramRefreshFlag: Boolean, TRUE if refreshFlag SHALL be set, FALSE otherwise   Either paramIccidValue or paramIsdpAidValue is passed as a parameter. |
| Details | IF paramIccidValue is provided Then  req EnableProfileRequest ::= {  profileIdentifier iccid : paramIccidValue,  refreshFlag paramRefreshFlag  }  Else  req EnableProfileRequest ::= {  profileIdentifier isdpAid : paramIsdpAidValue,  refreshFlag paramRefreshFlag  }  End if |

|  |  |
| --- | --- |
| Method | MTD\_ENABLE\_PROFILE\_MEP\_A1 |
| Description | Generates the ASN.1 EnableProfileRequest structure for MEP capable eUICCs configured with MEP-A1 according to the input parameters. |
| Parameter(s) | * paramIccidValue: The ICCID of the Profile to Enable (optional) * paramIsdpAidValue: The ISD-P AID of the Profile to Enable (optional) * paramRefreshFlag: Boolean, TRUE if refreshFlag SHALL be set, FALSE otherwise (optional) * paramTargetEsimPort: target eSIM port (mandatory)   Either paramIccidValue or paramIsdpAidValue is passed as a parameter.  paramRefreshFlag is required depending on the test scenario. |
| Details | If paramIccidValue is provided Then  req EnableProfileRequest ::= {  profileIdentifier iccid : paramIccidValue,  refreshFlag : paramRefreshFlag,  targetEsimPort : paramTargetEsimPort  }  Else  req EnableProfileRequest ::= {  profileIdentifier isdpAid : paramIsdpAidValue,  refreshFlag : paramRefreshFlag,  targetEsimPort : paramTargetEsimPort  } End if |

|  |  |
| --- | --- |
| Method | MTD\_GENERATE\_BPP |
| Description | Generate a BPP according to the input parameters. |
| Parameter(s) | * paramInitSC: The InitialiseSecureChannel request * paramConfISDP: The ConfigureISDP request (plain) * paramStoreMetadata: The StoreMetadata request (plain) * paramReplaceSessionKeys: The ReplaceSessionKeys request (plain) – Optional parameter * paramUPP: The Unprotected Profile Package to download |
| Details | Split the paramStoreMetadata in several segments of maximum 1008 bytes. Each Metadata segment is named <METADATA\_SEG> here after.  Split the paramUPP in several segments of maximum 1007 bytes. Each UPP segment named <UPP\_SEG> here after.  Create the following structure of data:  req BoundProfilePackage ::= {  paramInitSC,  firstSequenceOf87 {  0x87 <L> paramConfISDP  },  sequenceOf88 {  0x88 <L> <METADATA\_SEG>,  ...  0x88 <L> <METADATA\_SEG>  },  -- secondSequenceOf87 SHALL be set only if paramReplaceSessionKeys is provided  secondSequenceOf87 {  0x87 <L> paramReplaceSessionKeys  },  sequenceOf86 {  0x86 <L> <UPP\_SEG>,  ...  0x86 <L> <UPP\_SEG>  }  }  Use <OT\_SK\_S\_SM\_DP+\_ECKA> and <OTPK\_EUICC\_ECKA> in order to generate the <SHS>.  Concatenate #KEY\_TYPE, #KEY\_LENGTH, <L> #HOST\_ID and <L> #EID1 as SharedInfo.  Retrieve <S\_ENC>, <S\_MAC> and <S\_INIT\_MAC> across SHA-256 calculated from <SHS> and SharedInfo.  Encrypt paramConfISDP with <S\_ENC>.  Calculate and add a MAC to the tag 0x87 of firstSequenceOf87 by using <S\_MAC>.  Calculate and add a MAC to all tags 0x88 of sequenceOf88 by using <S\_MAC>.  If paramReplaceSessionKeys is provided Then  Encrypt paramReplaceSessionKeys with <S\_ENC>  Calculate and add a MAC to the tag 0x87 of secondSequenceOf87 by using <S\_MAC>.  End If  Encrypt all <UPP\_SEG> with <S\_ENC>, or <PPK\_ENC> if paramReplaceSessionKeys is provided.  Calculate and add a MAC to all tags 0x86 of sequenceOf86 by using <S\_MAC>, or <PPK\_MAC> (and <PPK\_INIT\_MAC> for the first tag) if paramReplaceSessionKeys is provided. |

|  |  |
| --- | --- |
| Method | MTD\_GENERATE\_HASHED\_CC |
| Description | Generate an Hashed Confirmation Code based on the Confirmation Code and the Transaction ID given in parameter. |
| Parameter(s) | * paramConfirmationCode: The Confirmation Code (plain) * paramTransactionId: The Transaction ID (plain) |
| Details | Generate a SHA-256 of the paramConfirmationCode.  Concatenate the obtained hash value with the paramTransactionId.  Generate and return a SHA-256 of these two concatenated elements. |

|  |  |
| --- | --- |
| Method | MTD\_GET\_PROFILE\_INFO |
| Description | Generate the ASN.1 ProfileInfoListRequest according to the input parameters. |
| Parameter(s) | * paramIccidValue: The ICCID of the Profile * paramIsdpAidValue: The ISD-P AID of the Profile   Either paramIccidValue or paramIsdpAidValue is passed as a parameter. |
| Details | If paramIccidValue is provided Then  req ProfileInfoListRequest::= {  searchCriteria iccid: paramIccidValue  }  Else if paramIsdpAidValue is provided then  req ProfileInfoListRequest::= {  searchCriteria isdpAid: paramIsdpAidValue  }  Else  req ProfileInfoListRequest::= {  }  End If |

|  |  |
| --- | --- |
| Method | MTD\_GET\_PROFILE\_INFO\_TAGLIST |
| Description | Generates the ASN.1 ProfileInfoListRequest to retrieve specified tags for all installed Profiles. |
| Parameter(s) | * paramTagList: list of target Tags to get Profile metadata of all installed profiles (mandatory) |
| Details | req ProfileInfoListRequest::= {  tagList: paramTagList } |

|  |  |
| --- | --- |
| Method | MTD\_LOAD\_RPM\_PKG\_REQ\_FOR\_LIST\_PROFILE\_INFO |
| Description | This Method can be used for creating LoadRpmPackageRequest for rpmPackage with single RPM command. |
| Parameter(s) | * paramTransactionId: The Transaction Id (Mandatory) * paramIccidValue: The ICCID within RPM Command * paramSmdpSignature3: The SM-DP+ Signature3 (Mandatory) * paramProfileOwnerOidValue: The ProfileOwnerOid for seachCriteria in ListProfileInfo if required * paramTagListValue: The TagList required for ListProfileInfo if required   Parameters paramTagListValue and paramProfileOwnerOidValue shall be passed and can be empty. |
| Details | If paramTagListValue is not present Then  If paramProfileOwnerOidValue is not present Then  loadRpmPkgReq LoadRpmPackageRequest ::= {  smdpSigned3 {  transactionId paramTransactionId,  rpmPackage {  {  continueOnFailure NULL,  rpmCommandDetails listProfileInfo : {  searchCriteria iccid : {paramIccidValue}  }  }  }  },  smdpSignature3 paramSmdpSignature3  }  Else  loadRpmPkgReq LoadRpmPackageRequest ::= {  smdpSigned3 {  transactionId paramTransactionId,  rpmPackage {  {  continueOnFailure NULL,  rpmCommandDetails listProfileInfo : {  searchCriteria profileOwnerOid : {paramProfileOwnerOidValue}  }  }  }  },  smdpSignature3 paramSmdpSignature3  }  Else  If paramProfileOwnerOidValue is not present Then  loadRpmPkgReq LoadRpmPackageRequest ::= {  smdpSigned3 {  transactionId paramTransactionId,  rpmPackage {  {  continueOnFailure NULL,  rpmCommandDetails listProfileInfo : {  searchCriteria iccid : {paramIccidValue},  tagList { paramTagListValue}  }  }  }  },  smdpSignature3 paramSmdpSignature3  }  Else if  loadRpmPkgReq LoadRpmPackageRequest ::= {  smdpSigned3 {  transactionId paramTransactionId,  rpmPackage {  {  continueOnFailure NULL,  rpmCommandDetails listProfileInfo : {  searchCriteria profileOwnerOid : {paramProfileOwnerOidValue},  tagList { paramTagListValue}  }  }  }  },  smdpSignature3 paramSmdpSignature3  }  End if |

|  |  |
| --- | --- |
| Method | MTD\_LOAD\_RPM\_PKG\_REQ\_MULT\_CMNDS |
| Description | This Method can be used to generate LoadRpmPackageRequest with rpmPackage containing multiple RPM commands. |
| Parameter(s) |  paramRpmComandList: The rpm commands list (Mandatory) |
| Details | loadRpmPkgReq LoadRpmPackageRequest ::= {  smdpSigned3 {  transactionId <S\_TRANSACTION\_ID>,  rpmPackage {  paramRpmComandList  }  }  },  smdpSignature3 <S\_SM\_DP+\_SIGNATURE3>  } |

|  |  |
| --- | --- |
| Method | MTD\_LOAD\_RPM\_PKG\_REQ\_SINGLE\_CMND |
| Description | This Method can be used for creating LoadRpmPackageRequest for rpmPackage with single RPM command. |
| Parameter(s) | * paramComandDetailsChoice: The rpmCommandDetails choice (Mandatory) * paramTransactionId: The Transaction Id (Mandatory) * paramIccidValue: The ICCID within RPM Command (Mandatory) * paramSmdpSignature3: The SM-DP+ Signature3 (Mandatory) * paramUpdateMetadataRequest: The UpdateMetadataRequest if required * paramDpiRpmValue: The dpiRpm within RPM Command if required   Parameters paramUpdateMetadataRequest and paramDpiRpmValue shall be passed and can be empty depend on the paramComandDetailsChoice. |
| Details | If paramComandDetailsChoice = enable or disable or delete Then  loadRpmPkgReq LoadRpmPackageRequest ::= {  smdpSigned3 {  transactionId paramTransactionId,  rpmPackage {  {  continueOnFailure NULL,  rpmCommandDetails paramComandDetailsChoice : {  iccid paramIccidValue  }  }  }  },  smdpSignature3 paramSmdpSignature3  }  Else if paramComandDetailsChoice = updateMetadata Then  loadRpmPkgReq LoadRpmPackageRequest ::= {  smdpSigned3 {  transactionId paramTransactionId,  rpmPackage {  {  continueOnFailure NULL,  rpmCommandDetails updateMetadata : {  iccid paramIccidValue  updateMetadataRequest {paramUpdateMetadataRequest}  }  }  }  },  smdpSignature3 paramSmdpSignature3  }  Else if paramComandDetailsChoice = contactPcmp Then  If paramDpiRpmValue is not present Then  loadRpmPkgReq LoadRpmPackageRequest ::= {  smdpSigned3 {  transactionId paramTransactionId,  rpmPackage {  {  continueOnFailure NULL,  rpmCommandDetails contactPcmp : {  iccid paramIccidValue  }  }  }  },  smdpSignature3 paramSmdpSignature3  }  Else  loadRpmPkgReq LoadRpmPackageRequest ::= {  smdpSigned3 {  transactionId paramTransactionId,  rpmPackage {  {  continueOnFailure NULL,  rpmCommandDetails contactPcmp : {  iccid paramIccidValue,  dpiRpm paramDpiRpmValue  }  }  }  },  smdpSignature3 paramSmdpSignature3  }  End if |

|  |  |
| --- | --- |
| Method | MTD\_REMOVE\_NOTIF |
| Description | Constructs the command data for RemoveNotificationFromList |
| Parameter(s) | * paramSeqNumber: the sequence number to be removed |
| Details | request NotificationSentRequest ::= {  seqNumber paramSeqNumber  } |

|  |  |
| --- | --- |
| Method | MTD\_REQ\_RPM\_CMND |
| Description | This Method can be used to generate a RpmCommand for given paramComandDetailsChoice and continueOnFailure flag.   * paramRpmComandList mentioned in MTD\_LOAD\_RPM\_PKG\_REQ\_MULT\_CMNDS can be generated |
| Parameter(s) | * paramComandDetailsChoice: The rpmCommandDetails choice (Mandatory) * paramIccidValue: The ICCID (Optional) * paramContinueOnFailureFlag: The Continue On Failure (Optional) * paramAdditional: The Additional parameters needed (Optional)   If the paramComandDetailsChoice is listProfileInfo paramIccidValue is optional.  If ContinueOnFailure shall be present set it as TRUE else it is not required.  paramAdditional:  If the paramComandDetailsChoice is updateMetadata UpdateMetadataRequest shall be included in paramAdditional.  If the paramComandDetailsChoice is listProfileInfo and need TagList to be specified it shall be included in paramAdditional |
| Details | If paramComandDetailsChoice = enable or disable or delete or contactPcmp Then  If paramContinueOnFailureFlag is present Then  {  continueOnFailure NULL,  rpmCommandDetails paramComandDetailsChoice : {  iccid paramIccidValue  }  }  Else  {  rpmCommandDetails paramComandDetailsChoice : {  iccid paramIccidValue  }  }  Else if paramComandDetailsChoice = listProfileInfo and paramAdditional is not present Then  If paramContinueOnFailureFlag is present Then  {  continueOnFailure NULL,  rpmCommandDetails listProfileInfo : {  searchCriteria iccid : { paramIccidValue }  }  }  Else  {  rpmCommandDetails listProfileInfo : {  searchCriteria iccid : { paramIccidvalue }  }  }  Else if paramComandDetailsChoice = listProfileInfo and paramAdditional is present Then  If paramContinueOnFailureFlag is present Then  {  continueOnFailure NULL,  rpmCommandDetails listProfileInfo : {  searchCriteria iccid : { paramIccidValue },  tagList paramAdditional  }  }  Else  {  rpmCommandDetails listProfileInfo : {  searchCriteria iccid : { paramIccidvalue },  tagList paramAdditional  }  }  Else – for updateMetadata  If paramContinueOnFailureFlag is present Then  {  continueOnFailure NULL,  rpmCommandDetails updateMetadata : {  iccid paramIccidvalue  updateMetadataRequest {paramAdditional}  }  }  Else  {  rpmCommandDetails updateMetadata : {  iccid paramIccidValue  updateMetadataRequest {paramAdditional}  }  }  End if |
|  |  |

|  |  |
| --- | --- |
| Method | MTD\_RES\_RPR\_CMND\_RESULT |
| Description | This Method can be used to generate the SuccessResult list or ErrorResult list resulted from multiple RPM commands.   * paramRpmCommandResultList mentioned in MTD\_RES\_RPR\_FOR\_MULT\_CMNDS can be generated. |
| Parameter(s) |  paramRprErrorMask: The RPR error mask as defined below. (Mandatory)   paramRpmCommandResultDataChoice: The rpmCommandDetails choice (Optional)   paramIccidValue: The ICCID within the RPM Command received (Optional)   paramProfileInfoValue: The Profile Info response (Optional)  paramRpmCommandResultError: The Rpm Command Result error (Optional)  paramRprErrorMask is defined as,  0 – OK result (RPM command successful)  1 - RpmCommandResultDataError is present  2 - RpmProcessingTerminated is present  paramRpmCommandResultDataChoice is not required for error code RpmProcessingTerminated.  Parameter paramIccidValue is mandatory if paramRpmCommandResultDataChoice is not listProfileInfoResult.  Parameter paramProfileInfoValue is mandatory if paramRprErrorMask = 0 (the successful responses) and if paramRpmCommandResultDataChoice is listProfileInfoResult.  Parameter paramRpmCommandResultError is mandatory if paramRprErrorMask > 0 (all error cases) and can be empty for paramRprErrorMask = 0 |
| Details | If paramRprErrorMask = 0 Then -- *OK response*  If paramRpmCommandResultDataChoice = enableResult or disableResult or deleteResult or updateMetadataResult Then  {  iccid paramIccidValue,  rpmCommandResultData paramRpmCommandResultDataChoice : {  paramRpmCommandResultDataChoice ok  }  }  Else if paramRpmCommandResultDataChoice = listProfileInfoResult Then    rpmCommandResultData listProfileInfoResult : {  profileInfoListOk : {paramProfileInfoValue }    Else if paramRpmCommandResultDataChoice = contactPcmpResult Then  {  iccid paramIccidValue,  rpmCommandResultData contactPcmpResult : {  contactPcmpResponseOk : {  pcmpAddress #TEST\_PCMP\_ADDRESS1 }  }  }  Else If paramRprErrorMask = 1 Then -- *rpmCommandResultDataError is present*  {  iccid paramIccidValue,  rpmCommandResultData paramRpmCommandResultDataChoice : {  paramRpmCommandResultDataChoice paramRpmCommandResultError  }  }  Else If paramRprErrorMask = 2 Then -- *rpmProcessingTerminated is present*  {  rpmCommandResultData rpmProcessingTerminated : {  paramRpmCommandResultError  }  }  End if |

|  |  |
| --- | --- |
| Method | MTD\_RES\_RPR\_FOR\_MULT\_CMNDS |
| Description | This Method can be used for verifying the rpmPackageResult resulted from multiple RPM commands. |
| Parameter(s) | paramRpmCommandResultList The Rpm Command Result list for multiple RPM commands (Mandatory) |
| Details | response LoadRpmPackageResult ::=  loadRpmPackageResultSigned : {  loadRpmPackageResultDataSigned {  transactionId <S\_TRANSACTION\_ID>,  notificationMetadata #NOTIF\_METADATA\_RPR,  smdpOid #S\_SM\_DP+\_OID,  finalResult rpmPackageExecutionResult : { paramRpmCommandResultList }  }  },  euiccSignRPR <EUICC\_SIGN\_RPR>  } |

|  |  |
| --- | --- |
| Method | MTD\_RES\_RPR\_FOR\_SINGLE\_CMND |
| Description | This Method can be used for verifying the rpmPackageResult resulted from a single RPM command. |
| Parameter(s) | * paramRpmCommandResultDataChoice: The rpmCommandDetails choice * paramTransactionId: The Transaction Id (Mandatory) * paramIccidValue: The ICCID within the RPM Command received * paramRprErrorMask: The RPR error mask as defined below. (Mandatory) * paramNotificationMetadata: The notification Metadata. * paramSmdpOid: The SM-DP+ OID. * paramProfileInfoValue: The Profile Info response * paramPcmpAddressValue: The pcmp Address. * paramRpmCommandResultError: The Rpm Command Result error   paramRprErrorMask is defined as,  0 – OK result (RPM command successful)  1 – RpmCommandResultDataError is present  2 – RpmProcessingTerminated is present  3 – LoadRpmPackageErrorCode is present  4 – LoadRpmPackageErrorCodeNotSigned is present  Parameters paramRpmCommandResultDataChoice and paramIccidValue are mandatory for paramRprErrorMask 0 and 1 only.  Parameters paramNotificationMetadata and paramSmdpOid are mandatory for all the cases except for paramRprErrorMask is 4 (loadRpmPackageErrorCodeNotSigned).  Parameters paramProfileInfoValue and paramPcmpAddressValue shall be passed and can be empty depend on the paramRpmCommandResultDataChoice for paramRprErrorMask is 0 (the successful responses).  Parameter paramRpmCommandResultError is mandatory if paramRprErrorMask > 0 (all error cases) and can be empty for paramRprErrorMask = 0. |
| Details | If paramRprErrorMask = 0 Then -- *OK response*  If paramRpmCommandResultDataChoice = enableResult or disableResult or deleteResult Then  response LoadRpmPackageResult ::=  loadRpmPackageResultSigned : {  loadRpmPackageResultDataSigned {  transactionId paramTransactionId,  notificationMetadata paramNotificationMetadata,  smdpOid paramSmdpOid,  finalResult rpmPackageExecutionResult : {  {  iccid paramIccidValue,  rpmCommandResultData paramRpmCommandResultDataChoice : {  paramRpmCommandResultDataChoice ok  }  }  }  },  euiccSignRPR <EUICC\_SIGN\_RPR>  }  Else if paramRpmCommandResultDataChoice = updateMetadataResult Then  response LoadRpmPackageResult ::=  loadRpmPackageResultSigned : {  loadRpmPackageResultDataSigned {  transactionId paramTransactionId,  notificationMetadata paramNotificationMetadata,  smdpOid paramSmdpOid,  finalResult rpmPackageExecutionResult : {  {  iccid paramIccidValue,  rpmCommandResultData updateMetadataResult : ok  }  }  },  euiccSignRPR <EUICC\_SIGN\_RPR>  }  Else if paramRpmCommandResultDataChoice = listProfileInfoResult Then  response LoadRpmPackageResult ::=  loadRpmPackageResultSigned : {  loadRpmPackageResultDataSigned {  transactionId paramTransactionId,  notificationMetadata paramNotificationMetadata,  smdpOid paramSmdpOid,  finalResult rpmPackageExecutionResult : {  {  rpmCommandResultData listProfileInfoResult : {  profileInfoListOk : {paramProfileInfoValue }  }  }  }  },  euiccSignRPR <EUICC\_SIGN\_RPR>  }  Else if paramRpmCommandResultDataChoice = contactPcmpResult Then  response LoadRpmPackageResult ::=  loadRpmPackageResultSigned : {  loadRpmPackageResultDataSigned {  transactionId paramTransactionId,  notificationMetadata paramNotificationMetadata,  smdpOid paramSmdpOid,  finalResult rpmPackageExecutionResult : {  {  iccid paramIccidValue,  rpmCommandResultData contactPcmpResult : {  contactPcmpResponseOk : {  pcmpAddress paramPcmpAddressValue }  }  }  }  },  euiccSignRPR <EUICC\_SIGN\_RPR>  }  Else If paramRprErrorMask = 1 Then -- *rpmCommandResultDataError is present*  If paramRpmCommandResultDataChoice = listProfileInfoResult Then  response LoadRpmPackageResult ::=  loadRpmPackageResultSigned : {  loadRpmPackageResultDataSigned {  transactionId paramTransactionId,  notificationMetadata paramNotificationMetadata,  smdpOid paramSmdpOid,  finalResult rpmPackageExecutionResult : {  {  rpmCommandResultData listProfileInfoResult : {  profileInfoListError : { paramRpmCommandResultError }  }  }  }  },  euiccSignRPR <EUICC\_SIGN\_RPR>  }  Else if paramRpmCommandResultDataChoice = contactPcmpResult Then  response LoadRpmPackageResult ::=  loadRpmPackageResultSigned : {  loadRpmPackageResultDataSigned {  transactionId paramTransactionId,  notificationMetadata paramNotificationMetadata,  smdpOid paramSmdpOid,  finalResult rpmPackageExecutionResult : {  {  iccid paramIccidValue,  rpmCommandResultData contactPcmpResult: {  contactPcmpResponseError: { paramRpmCommandResultError }  }  }  }  },  euiccSignRPR <EUICC\_SIGN\_RPR>  }  Else if paramRpmCommandResultDataChoice = updateMetadataResult Then  response LoadRpmPackageResult ::=  loadRpmPackageResultSigned : {  loadRpmPackageResultDataSigned {  transactionId paramTransactionId,  notificationMetadata paramNotificationMetadata,  smdpOid paramSmdpOid,  finalResult rpmPackageExecutionResult : {  {  iccid paramIccidValue,  rpmCommandResultData updateMetadataResult:  paramRpmCommandResultError  }  }  },  euiccSignRPR <EUICC\_SIGN\_RPR>  }  Else  response LoadRpmPackageResult ::=  loadRpmPackageResultSigned : {  loadRpmPackageResultDataSigned {  transactionId paramTransactionId,  notificationMetadata paramNotificationMetadata,  smdpOid paramSmdpOid,  finalResult rpmPackageExecutionResult : {  {  iccid paramIccidValue,  rpmCommandResultData paramRpmCommandResultDataChoice : {  paramRpmCommandResultDataChoice paramRpmCommandResultError  }  }  }  },  euiccSignRPR <EUICC\_SIGN\_RPR>  }  Else If paramRprErrorMask = 2 Then -- *rpmProcessingTerminated is present*  response LoadRpmPackageResult ::=  loadRpmPackageResultSigned : {  loadRpmPackageResultDataSigned {  transactionId paramTransactionId,  notificationMetadata paramNotificationMetadata,  smdpOid paramSmdpOid,  finalResult rpmPackageExecutionResult : {  {  rpmCommandResultData rpmProcessingTerminated : {  paramRpmCommandResultError  }  }  }  },  euiccSignRPR <EUICC\_SIGN\_RPR>  }  Else If paramRprErrorMask = 3 Then -- *loadRpmPackageErrorCode is present*  response LoadRpmPackageResult ::= {  loadRpmPackageResultSigned : {  loadRpmPackageResultDataSigned {  transactionId paramTransactionId,  notificationMetadata paramNotificationMetadata,  smdpOid paramSmdpOid,  finalResult loadRpmPackageErrorCode : {  paramRpmCommandResultError }  },  euiccSignRPR <EUICC\_SIGN\_RPR>  }  Else If paramRprErrorMask = 4 Then -- loadRpmPackageErrorCodeNotSigned is present  response LoadRpmPackageResult ::=  loadRpmPackageResultNotSigned : {  transactionId paramTransactionId,  loadRpmPackageErrorCodeNotSigned paramRpmCommandResultError  }  End if |

|  |  |
| --- | --- |
| Method | MTD\_RETRIEVE\_NOTIF\_SEQ\_NUM |
| Description | Constructs the command data for RetrieveNotificationsList filtered by sequence number |
| Parameter(s) | * paramSeqNumber: the sequence number to be retrieved |
| Details | request RetrieveNotificationsListRequest ::= {  searchCriteria seqNumber paramSeqNumber } |

|  |  |
| --- | --- |
| Method | MTD\_SELECT |
| Description | Generates the SELECT command as defined in GlobalPlatform Card Specification [6]. |
| Parameter(s) |  paramAID: the AID to select |
| Details | - CLA = 0x or 4x (x = <CHANNEL\_NUMBER>)  - INS = A4  - P1 = 04  - P2 = 00  - LC = <L>  - paramAID  - LE = 00 |

|  |  |
| --- | --- |
| Method | MTD\_SEND\_SMS\_PP |
| Description | Generate and send an envelope SMS-PP download to the MNO-SD |
| Parameter(s) | * paramApdusList: the list of APDUs (plain) to send |
| Details | Generate and send the following envelope:  80 C2 00 00 <L>  D1 <L>  02 02 83 81 -- Device identity Tag  06 07 91 33 86 09 40 00 F0 -- Address Tag (TON/NPI/..)  0B <L> -- SMS TPDU  44 -- SMS-DELIVER  05 85 02 13 F2 -- TP-Originating-Address  7F -- TP-Protocol-Identifier  F6 -- TP-Data-Coding-Scheme  71 30 12 41 55 74 40 -- TP-Service-Centre-Time-Stamp  <L> -- TP-User-Data-Length  02 -- User-Data-Header-Length  70 -- IEIa  00 -- IEIDLa  <L> -- Command Packet Length (2 bytes)  <L> -- Command Header Length (1 byte)  12 21 -- SPI  00 -- KIC  15 –- KID (SCP80 Keyset version 0x01 in Triple DES)  B2 01 00 –- MNO-SD TAR  <MNO\_SCP80\_COUNTER>  00 -- Padding Counter  <CC> -- Cryptographic checksum  <C\_APDUS\_SCRIPT> -- Command APDUs script  <C\_APDUS\_SCRIPT> SHALL contain the paramApdusList (i.e. each APDU is named <APDU1>; <APDU2>; …; <APDUn> here after) formatted as an expanded structure with definite length as defined in ETSI TS 102 226 [14]:  AA <L>  22 <L> <APDU1>  22 <L> <APDU2>  ...  22 <L> <APDUn>  The Cryptographic checksum <CC> SHALL be generated in Triple DES (outer-CBC mode using two different keys) with the #MNO\_SCP80\_AUTH\_KEY as defined in ETSI TS 102 225 [13].  If the command packet length is higher than 140 bytes, it SHALL be sent over several envelopes: SMS concatenation as defined in 3GPP TS 23.040 [22] SHALL be used. |

|  |  |
| --- | --- |
| Method | MTD\_STORE\_DATA |
| Description | Generates the STORE DATA command (Case 4) as defined in GlobalPlatform Card Specification [6]. |
| Parameter(s) | * paramCommandData: the command data |
| Details | - CLA = 8x or Cx (x = <CHANNEL\_NUMBER>)  - INS = E2  - P1 = 91  - P2 = 00  - LC = <L>  - paramCommandData  - LE = 00 |

|  |  |
| --- | --- |
| Method | MTD\_STORE\_DATA\_Case3 |
| Description | Generates the STORE DATA command (Case3) as defined in GlobalPlatform Card Specification [6]. |
| Parameter(s) | * paramCommandData: the command data |
| Details | - CLA = 8x or Cx (x = <CHANNEL\_NUMBER>)  - INS = E2  - P1 = 90  - P2 = 00  - LC = <L>  - paramCommandData |

|  |  |
| --- | --- |
| Method | MTD\_STORE\_DATA\_SCRIPT |
| Description | Generate (multiple) STORE DATA command(s) by breaking the data into smaller components (if needed) for transmission. |
| Parameter(s) | * paramTLVDataToTransmit: TLVs array or single TLV to transfer to the eUICC * paramCase4Command (optional parameter, default value = TRUE): TRUE if the APDU is a Case 4 command, FALSE if the APDU is a Case 3 command |
| Details | For each element of paramTLVDataToTransmit  If the size of the element is greater than 255 bytes, split the element in several blocks of  255 bytes. The last block MAY be shorter. Each block is named <DATA\_SUB\_PART> here after.  If the element is up to 255 bytes, <DATA\_SUB\_PART> contains the value of the element.  The bit b1 of P1 in the STORE DATA commands is named <B1\_P1> here after and is  defined as below:  If paramCase4Command = TRUE Then  <B1\_P1> = 1  Else  <B1\_P1> = 0  End If  Set <STORE\_DATA\_BLOCK\_NUM> to 0  For each <DATA\_SUB\_PART>  If <DATA\_SUB\_PART> is an intermediate part, generate the following STORE DATA:  - CLA = 8x or Cx (x = <CHANNEL\_NUMBER>)  - INS = E2  - P1 = 1x (x = <B1\_P1>)  - P2 = <STORE\_DATA\_BLOCK\_NUM>  - LC = <L>  - <DATA\_SUB\_PART>  - LE = 00 –- present only if paramCase4Command = TRUE  If <DATA\_SUB\_PART> is the last part, generate the following STORE DATA:  - CLA = 8x or Cx (x = <CHANNEL\_NUMBER>)  - INS = E2  - P1 = 9x (x = <B1\_P1>)  - P2 = <STORE\_DATA\_BLOCK\_NUM>  - LC = <L>  - <DATA\_SUB\_PART>  - LE = 00 –- present only if paramCase4Command = TRUE  Increase the <STORE\_DATA\_BLOCK\_NUM> by 1  End  End |

C.2 Procedures

|  |  |  |  |
| --- | --- | --- | --- |
|  | Procedure | PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP | |
|  | Description | Configures LSIs to support MEP with the MEP mode and number of LSIs as needed by the test sequence. | |
|  | Parameters | * paramHighestLsi: The highest LSI proposed by the scenario. * paramLsiOptions: LSI options supported by the scenario. * paramMepModePriority: MEP mode priority order supported by the scenario. * paramMaxLsiNum: The maximum number of LSIs for Enabled Profiles   paramMepModePriority: up to 3 bytes to represent the priority order.  e.g.  ‘010203’ indicates priority order as ‘MEP-A1, MEP-A2, MEP-B’.  ‘0301’ indicates priority order as ‘MEP-B, MEP-A1’, and not supporting MEP-A2.  ‘01’ indicates S\_Device supports only MEP-A1, not supporting MEP-A2 or MEP-B  (Forcing eUICC to select MEP-A1 in case it supports any other MEP mode as well in addition to MEP-A1). | |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_Device → eUICC | send ‘LSI support’ in PPS2 |  |
| 2 | S\_Device → eUICC | send MANAGE\_LSI(Configure LSI) with the following tags:   * ‘Highest LSI proposed by the terminal’ set to paramHighestLsi. * ‘LSI options supported by the terminal’ set to paramLsiOptions. * ‘MEP mode(s) supported by the S\_Device in the order of priority’ set to   paramMepModePriority   * ‘Maximum number of LSIs for Enabled Profiles of the S\_Device’ set to paramMaxLsiNum; | SW=9000  Parse response and :   * If ‘LSI Options’ is present Then   + extract ‘LSI Options’ as <MEP\_LSI\_OPTIONS> * If ‘LSI Options’ is not present Then   + <MEP\_LSI\_OPTIONS> is 0x00 and * extract ‘Jointly supported MEP mode’ as <MEP\_MODE> * extract ‘Jointly supported maximum number of LSIs for Enabled Profiles’ as <MEP\_MAX\_LSIS> |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Procedure | PROC\_EUICC\_INITIALIZATION\_SEQUENCE | |
|  | Description | Initialize communication between the S\_Device and the eUICC. | |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_Device → eUICC | RESET | ATR present |
| 2 | S\_Device → eUICC | [SELECT\_MF] | FCP Template present  SW=0x9000 |
| 3 | S\_Device → eUICC | [TERMINAL\_CAPABILITY\_LPAd] | SW=0x9000 |
| 4 | S\_Device → eUICC | [TERMINAL\_PROFILE] | Toolkit initialization THEN SW=0x9000 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Procedure | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_Enterprise | |
|  | Description | Initialize communication between the S\_Device and the eUICC. | |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_Device → eUICC | RESET | ATR present |
| 2 | S\_Device → eUICC | [SELECT\_MF] | FCP Template present  SW=0x9000 |
| 3 | S\_Device → eUICC | [TERMINAL\_CAPABILITY\_Enterprise] | SW=0x9000 |
| 4 | S\_Device → eUICC | [TERMINAL\_PROFILE] | Toolkit initialization THEN SW=0x9000 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Procedure | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged | |
|  | Description | Initialize communication between the S\_Device and the eUICC. | |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_Device → eUICC | RESET | ATR returned by eUICC |
| 2 | S\_Device → eUICC | [SELECT\_MF] | FCP Template present  SW=0x9000 |
| 3 | S\_Device → eUICC | [TERMINAL\_CAPABILITY\_LPAd] | SW=0x9000 |
| 4 | S\_Device → eUICC | [TERMINAL\_PROFILE\_eUICCProfileStateChanged] | Toolkit initialization THEN SW=0x9000 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Procedure | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP | |
|  | Description | Initialize communication between the S\_Device and the MEP capable eUICC after configuring 3 LSIs using the MANGE LSI (Configure LSI) command. It performs MEP eUICC initialization procedure for 3 LSIs configured.  Expects TCs to have the steps for RESET and the PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP before calling this PROC. | |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_Device | PROC\_MEP\_LSI\_MULTIPLEXING(0) |  |
| 2 | S\_Device → eUICC | [SELECT\_MF] | FCP Template present  SW=0x9000 |
| 3 | S\_Device → eUICC | [TERMINAL\_CAPABILITY\_LPAd] | SW=0x9000 |
| 4 | S\_Device → eUICC | [TERMINAL\_PROFILE\_LSI\_COMMAND\_eUICCProfileStateChanged] | Toolkit initialization THEN SW=0x9000 |
| 5 | S\_Device | PROC\_MEP\_LSI\_MULTIPLEXING(1) |  |
| Repeat Step 2 to 4 | | | |
| 6 | S\_Device | PROC\_MEP\_LSI\_MULTIPLEXING(0) |  |
| IF (NOT(<MEP\_MODE>=MEP-B)) | | | |
| 7 | S\_Device | PROC\_MEP\_LSI\_MULTIPLEXING(2) |  |
| 8 | Repeat Step 4 to 6 | | |
| 9 | S\_Device | PROC\_MEP\_LSI\_MULTIPLEXING(0) |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Procedure | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP\_EN\_DS\_FIRST\_PROFILE | |
|  | Description | Initialize communication between the S\_Device and the MEP capable eUICC after enable or disable the profile operation for the 1st profile.  NOTE: 1st profile means the profile assigned to LSI-1 for MEP-A1 or MEP-A2, or the profile assigned to LSI-0 for MEP-B | |
| Step | Direction | Sequence / Description | Expected result |
| IF (NOT(<MEP\_MODE> = MEP-B)) | | | |
| 1 | S\_Device | PROC\_MEP\_LSI\_MULTIPLEXING(1) |  |
| ENDIF | | | |
| IF (<MEP\_MODE> = MEP-B) | | | |
| 2 | S\_Device | PROC\_MEP\_LSI\_MULTIPLEXING(0) |  |
| ENDIF | | | |
| 3 | S\_Device → eUICC | [SELECT\_MF] | FCP Template present  SW=0x9000 |
| 4 | S\_Device → eUICC | [TERMINAL\_CAPABILITY\_LPAd] | SW=0x9000 |
| 5 | S\_Device → eUICC | [TERMINAL\_PROFILE\_LSI\_COMMAND\_eUICCProfileStateChanged] | Toolkit initialization THEN SW=0x9000 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Procedure | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_eUICCProfileStateChanged\_MEP\_EN\_DS\_SECOND\_PROFILE | |
|  | Description | Initialize communication between the S\_Device and the eUICC after enable or disable the profile operation for the 2nd profile.  NOTE: 2nd profile means the profile assigned to LSI-2 for MEP-A1 or MEP-A2, or the profile assigned to LSI-1 for MEP-B | |
| Step | Direction | Sequence / Description | Expected result |
| IF (NOT(<MEP\_MODE> = MEP-B)) | | | |
| 1 | S\_Device | PROC\_MEP\_LSI\_MULTIPLEXING(2) |  |
| ENDIF | | | |
| IF (<MEP\_MODE> = MEP-B) | | | |
| 2 | S\_Device | PROC\_MEP\_LSI\_MULTIPLEXING(1) |  |
| ENDIF | | | |
| 3 | S\_Device → eUICC | [SELECT\_MF] | FCP Template present  SW=0x9000 |
| 4 | S\_Device → eUICC | [TERMINAL\_CAPABILITY\_LPAd] | SW=0x9000 |
| 5 | S\_Device → eUICC | [TERMINAL\_PROFILE\_LSI\_COMMAND\_eUICCProfileStateChanged] | Toolkit initialization THEN SW=0x9000 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Procedure | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_LPA\_Alerting | |
|  | Description | Initialize communication between the S\_Device and the eUICC. | |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_Device → eUICC | RESET | ATR present |
| 2 | S\_Device → eUICC | [SELECT\_MF] | FCP Template present  SW=0x9000 |
| 3 | S\_Device → eUICC | [TERMINAL\_CAPABILITY\_LPA\_Alerting] | SW=0x9000 |
| 4 | S\_Device → eUICC | [TERMINAL\_PROFILE] | Toolkit initialization THEN SW=0x9000 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Procedure | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP | |
|  | Description | Initialize communication between the S\_Device and the MEP capable eUICC after using the MANAGE LSI(Configure LSI) command to configure enough LSIs to host two simultaneously Enabled Profiles. It performs MEP eUICC initialization procedure for all LSIs configured.  Expects TCs to have the steps for RESET and the PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP before calling this PROC. | |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_Device | PROC\_MEP\_LSI\_MULTIPLEXING(0) |  |
| 2 | S\_Device → eUICC | [SELECT\_MF] | FCP Template present  SW=0x9000 |
| 3 | S\_Device → eUICC | [TERMINAL\_CAPABILITY\_LPAd] | SW=0x9000 |
| 4 | S\_Device → eUICC | [TERMINAL\_PROFILE\_LSI\_COMMAND] | Toolkit initialization THEN SW=0x9000 |
| 5 | S\_Device | PROC\_MEP\_LSI\_MULTIPLEXING(1) |  |
| 6 | Repeat steps 2-4 | | |
| IF (NOT(<MEP\_MODE> = MEP-B)) | | | |
| 7 | S\_Device | PROC\_MEP\_LSI\_MULTIPLEXING(2) |  |
| 8 | Repeat steps 2-4 | | |
| ENDIF | | | |
| 9 | S\_Device | PROC\_MEP\_LSI\_MULTIPLEXING(0) |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Procedure | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_FIRST\_PROFILE | |
|  | Description | Initialize communication between the S\_Device and the MEP capable eUICC after enable or disable the profile operation for the 1st profile.  NOTE: 1st profile means the profile assigned to LSI-1 for MEP-A1 or MEP-A2, or the profile assigned to LSI-0 for MEP-B | |
| Step | Direction | Sequence / Description | Expected result |
| IF (NOT(<MEP\_MODE> = MEP-B)) | | | |
| 1 | S\_Device | PROC\_MEP\_LSI\_MULTIPLEXING(1) |  |
| ENDIF | | | |
| IF (<MEP\_MODE> = MEP-B) | | | |
| 2 | S\_Device | PROC\_MEP\_LSI\_MULTIPLEXING(0) |  |
| ENDIF | | | |
| 3 | S\_Device → eUICC | [SELECT\_MF] | FCP Template present  SW=0x9000 |
| 4 | S\_Device → eUICC | [TERMINAL\_CAPABILITY\_LPAd] | SW=0x9000 |
| 5 | S\_Device → eUICC | [TERMINAL\_PROFILE\_LSI\_COMMAND] | Toolkit initialization THEN SW=0x9000 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Procedure | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_EN\_DS\_SECOND\_PROFILE | |
|  | Description | Initialize communication between the S\_Device and the eUICC after enable or disable the profile operation for the 2nd profile.  NOTE: 2nd profile means the profile assigned to LSI-2 for MEP-A1 or MEP-A2, or the profile assigned to LSI-1 for MEP-B | |
| Step | Direction | Sequence / Description | Expected result |
| IF (NOT(<MEP\_MODE> = MEP-B)) | | | |
| 1 | S\_Device | PROC\_MEP\_LSI\_MULTIPLEXING(2) |  |
| ENDIF | | | |
| IF (<MEP\_MODE> = MEP-B) | | | |
| 2 | S\_Device | PROC\_MEP\_LSI\_MULTIPLEXING(1) |  |
| ENDIF | | | |
| 3 | S\_Device → eUICC | [SELECT\_MF] | FCP Template present  SW=0x9000 |
| 4 | S\_Device → eUICC | [TERMINAL\_CAPABILITY\_LPAd] | SW=0x9000 |
| 5 | S\_Device → eUICC | [TERMINAL\_PROFILE\_LSI\_COMMAND] | Toolkit initialization THEN SW=0x9000 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Procedure | PROC\_EUICC\_INITIALIZATION\_SEQUENCE\_MEP\_ONE\_LSI\_FOR\_ENABLED\_PROFILE | |
|  | Description | Initialises communication between the S\_Device and the MEP capable eUICC with only one LSI out of the two or more LSIs configured for Enabled Profiles by a previous MANAGE LSI(Configure LSI) command.  Expects TCs to have the steps for RESET and the PROC\_EUICC\_CONFIGURE\_LSIS\_FOR\_MEP before calling this PROC. | |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_Device | PROC\_MEP\_LSI\_MULTIPLEXING(0) |  |
| 2 | S\_Device → eUICC | [SELECT\_MF] | FCP Template present  SW=0x9000 |
| 3 | S\_Device → eUICC | [TERMINAL\_CAPABILITY\_LPAd] | SW=0x9000 |
| 4 | S\_Device → eUICC | [TERMINAL\_PROFILE\_LSI\_COMMAND] | Toolkit initialization THEN SW=0x9000 |
| IF (NOT(<MEP\_MODE> = MEP-B)) | | | |
| 5 | S\_Device | PROC\_MEP\_LSI\_MULTIPLEXING(1) |  |
| Repeat steps 2-4 | | | |
| ENDIF | | | |
| 6 | S\_Device | PROC\_MEP\_LSI\_MULTIPLEXING(0) |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Procedure | PROC\_MEP\_LSI\_MULTIPLEXING | |
|  | Description | Selects the LSI for the given eSIM Port number to be used for the next APDUs on an eUICC configured for MEP MEP-capable eUICC.  The test tool SHALL use the LSI selection method indicated by the EUM in #IUT\_EUICC\_MULTIPLEXING\_LSI\_INDICATION.  All subsequent APDUs are sent on the indicated paramPortForNextCommand until the next call to this procedure.  This procedure does nothing if the current command Port is already the one specified in the parameter paramPortForNextCommand. | |
|  | Parameters | paramCommandPort: LSI number on which to send the subsequent APDUs | |
| Step | Direction | Sequence / Description | Expected result |
| IF (#IUT\_EUICC\_MULTIPLEXING\_LSI\_INDICATION = 'MANAGE LSI(Select LSI)') | | | |
| 1 | S\_Device → eUICC | Send 'MANAGE LSI(Select LSI) with the LSI Number set to paramPortForNextCommand | SW=9000 |
| 2 | S\_Device → eUICC | Send the subsequent commands specified by the Test Sequence unchanged, until the next call to this procedure. | |
| ENDIF | | | |
| IF (#IUT\_EUICC\_MULTIPLEXING\_LSI\_INDICATION = ' T=1 + NAD byte') | | | |
| 3 | S\_Device → eUICC | Construct the transport for the subsequent commands specified by the Test Sequence by setting the LSI number paramPortForNextCommand in the NAD byte, until the next call to this procedure. | |
| ENDIF | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Procedure | | PROC\_MEP\_REFRESH\_EN\_DS | | |
|  | Description | | Handles the REFRESH on an LSI and simulates the device processing it after enabling or disabling profiles. | | |
|  | Parameters | | * paramTargetEsimPort: target eSIM port for ES10 calls (mandatory for MEP-A1) * paramRefreshMode: Refresh mode “eUICC Profile State Change" or “UICC Reset”(mandatory) | | |
| Step | Direction | | Sequence / Description | | Expected result |
| IF (<MEP\_MODE> = ‘MEP-A1’) | | | | | |
| NOTE: in this case the enable or disable command was sent on port 0, and the resulting SW was 91YY | | | | | |
| 1 | S\_Device → eUICC | | FETCH ‘YY’ | | LSI COMMAND Proactive Command  Extract ‘Action’ as <LSI\_COMMAND\_ACTION> from the command qualifier  Verify that <LSI\_COMMAND\_ACTION> is equal to “Proactive session request”  Verify that ‘LSI numbers’ contains a single number and extract it as <LSI\_NUMBER>  Verify that <LSI\_NUMBER> is equal to paramTargetEsimPort |
| 2 | S\_Device → eUICC | | [TERMINAL RESPONSE] | |  |
| 3 | PROC\_MEP\_LSI\_MULTIPLEXING(paramTargetEsimPort) | | | | |
| 4 | S\_Device → eUICC | FETCH ‘Le=00’ | | REFRESH Proactive Command  Verify that the Refresh Mode in the Proactive Command is equal to paramRefreshMode | |
| ENDIF | | | | | |
| NOTE: if <MEP\_MODE> = ‘MEP-B’, the enable or disable command was sent on the target port, and the resulting SW was 91XX | | | | | |
| 5 | S\_Device → eUICC | | FETCH ‘XX’ | | REFRESH Proactive Command  Verify that the Refresh Mode in the Proactive Command is equal to paramRefreshMode |
| ENDIF | | | | | |
| IF (paramRefreshMode = “eUICC Profile State Change") | | | | | |
| 6 | S\_Device → eUICC | | [TERMINAL RESPONSE] | |  |
| ENDIF | | | | | |
| IF (paramRefreshMode = “UICC Reset”) | | | | | |
| 7 | S\_Device → eUICC | | MANAGE\_LSI(Reset LSE, paramTargetEsimPort) | |  |
| ENDIF | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Procedure | PROC\_OPEN\_LOGICAL\_CHANNEL\_AND\_SELECT\_ISDR | |
|  | Description | The LPAd opens a logical channel and selects the ISD-R.  Since the S\_LPAd may have to do that on more than one LSI, the S\_LPAd SHALL record one value of the <CHANNEL NUMBER> per LSI. | |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | [MANAGE\_CHANNEL\_OPEN] | Extract the <CHANNEL\_NUMBER> from response data  SW=0x9000 |
| 2 | S\_LPAd → eUICC | MTD\_SELECT(#ISD\_R\_AID) | SW=0x9000 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Procedure | PROC\_VERIFY\_SESSION\_IS\_CANCELLED | |
|  | Description | Verify that the RSP session identified by the TransactionID <S\_TRANSACTION\_ID> has been cancelled by the eUICC (i.e. Common Mutual Authentication and Profile Download procedures SHALL be rejected as long as no GetEUICCChallenge has been requested). | |
| Step | Direction | Sequence / Description | Expected result |
| 1 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  #PREP\_DOWNLOAD\_NO\_CC) | #R\_PREP\_DOWN\_NO\_SESSION  SW=0x9000  The transactionId returned in the response SHALL not be checked (any value SHALL be accepted). |
| 3 | S\_LPAd → eUICC | MTD\_STORE\_DATA\_SCRIPT(  MTD\_AUTHENTICATE\_SMDP(  #TEST\_DP\_ADDRESS1,  <S\_SMDP\_CHALLENGE>,  #CTX\_PARAMS1,  <S\_SMDP\_SIGNATURE1>,  #CERT\_S\_SM\_DPauth\_SIG,  NO\_PARAM,  #CRL\_LIST,  FALSE  )  ) | #R\_AUTH\_SERVER\_NO\_SESSION  SW = 0x9000  The transactionId returned in the response SHALL not be checked (any value SHALL be accepted). |

Annex D Commands And Responses

D.1 ES8+ Requests And Responses

D.1.1 ES8+ Requests

| Name | Content |
| --- | --- |
| CONF\_ISDP\_EMPTY | req ConfigureISDPRequest ::={} |
| CONF\_ISDP\_MAX\_LENGTH | req ConfigureISDPRequest ::={  dpProprietaryData { *-- size=128 bytes*  dpOid #S\_SM\_DP+\_OID,  additionalSmdpData #ADDITIONAL\_SMDP\_DATA\_MAX\_LENGTH  }  }  -- NOTE: Instead of  DpProprietaryData ::= SEQUENCE {  dpOid OBJECT IDENTIFIER  -- additional data objects defined by the  -- SM-DP+ MAY follow  }  -- the following structure is used to test the  -- DpProprietaryData size:  DpProprietaryData ::= SEQUENCE {  dpOid OBJECT IDENTIFIER,  additionalSmdpData OCTET STRING OPTIONAL  } |
| CONF\_ISDP\_PROF1 | req ConfigureISDPRequest ::={  dpProprietaryData {  dpOid #S\_SM\_DP+\_OID  }  } |
| CONF\_ISDP\_PROF2 | req ConfigureISDPRequest ::={  dpProprietaryData {  dpOid #S\_SM\_DP+\_OID  }  } |
| CONF\_ISDP\_SIZE\_EXCEEDED | req ConfigureISDPRequest ::={  dpProprietaryData { *-- size=129 bytes*  dpOid #S\_SM\_DP+\_OID,  additionalSmdpData  #ADDITIONAL\_SMDP\_DATA\_EXCEEDED\_MAX  }  }  -- NOTE: Instead of  DpProprietaryData ::= SEQUENCE {  dpOid OBJECT IDENTIFIER  -- additional data objects defined by the  -- SM-DP+ MAY follow  }  -- the following structure is used to test the  -- DpProprietaryData size:  DpProprietaryData ::= SEQUENCE {  dpOid OBJECT IDENTIFIER,  additionalSmdpData OCTET STRING OPTIONAL  } |
| FULL\_METADATA | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  notificationConfigurationInfo {  { profileManagementOperation {  notificationInstall,  notificationLocalEnable,  notificationLocalDisable,  notificationLocalDelete  },  notificationAddress #TEST\_DP\_ADDRESS1  }  },  profileOwner {  mccMnc #MCC\_MNC1  },  profilePolicyRules {ppr1}  } |
| INIT\_SC\_INVALID\_CRT | req InitialiseSecureChannelRequest ::={  remoteOpId #REMOTE\_OP\_ID\_INSTALL,  transactionId <S\_TRANSACTION\_ID>,  controlRefTemplate {  keyType #INVALID\_KEY\_TYPE,  keyLen #KEY\_LENGTH,  hostId #HOST\_ID  },  smdpOtpk <OTPK\_S\_SM\_DP+\_ECKA>,  smdpSign <S\_SM\_DP+\_SIGN>  } |
| INIT\_SC\_INVALID\_OP\_ID | req InitialiseSecureChannelRequest ::={  remoteOpId #INVALID\_REMOTE\_OP\_ID,  transactionId <S\_TRANSACTION\_ID>,  controlRefTemplate {  keyType #KEY\_TYPE,  keyLen #KEY\_LENGTH,  hostId #HOST\_ID  },  smdpOtpk <OTPK\_S\_SM\_DP+\_ECKA>,  smdpSign <S\_SM\_DP+\_SIGN>  } |
| INIT\_SC\_INVALID\_SIGN | req InitialiseSecureChannelRequest ::={  remoteOpId #REMOTE\_OP\_ID\_INSTALL,  transactionId <S\_TRANSACTION\_ID>,  controlRefTemplate {  keyType #KEY\_TYPE,  keyLen #KEY\_LENGTH,  hostId #HOST\_ID  },  smdpOtpk <OTPK\_S\_SM\_DP+\_ECKA>,  smdpSign <S\_SM\_DP+\_SIGN>  }  The <S\_SM\_DP+\_SIGN> SHALL NOT be computed using the #SK\_S\_SM\_DPpb\_SIG but SHALL have the same length as for a valid signature. |
| INIT\_SC\_INVALID\_TRANS\_ID | req InitialiseSecureChannelRequest ::={  remoteOpId #REMOTE\_OP\_ID\_INSTALL,  transactionId <INVALID\_TRANSACTION\_ID>,  controlRefTemplate {  keyType #KEY\_TYPE,  keyLen #KEY\_LENGTH,  hostId #HOST\_ID  },  smdpOtpk <OTPK\_S\_SM\_DP+\_ECKA>,  smdpSign <S\_SM\_DP+\_SIGN>  } |
| METADATA\_ICCID\_MISMATCH | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF2,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1  } |
| METADATA\_MCCMNC\_MISMATCH | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  profileOwner {  mccMnc #MCC\_MNC2  },  profilePolicyRules {ppr2}  } |
| METADATA\_NO\_CLASS | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  notificationConfigurationInfo {  { profileManagementOperation {  notificationInstall,  notificationLocalEnable,  notificationLocalDisable,  notificationLocalDelete  },  notificationAddress #TEST\_DP\_ADDRESS1  }  }  } |
| METADATA\_OP\_PROF1 | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  notificationConfigurationInfo {  { profileManagementOperation {  notificationInstall,  notificationLocalEnable,  notificationLocalDisable,  notificationLocalDelete  },  notificationAddress #TEST\_DP\_ADDRESS1  }  },  profileOwner {  mccMnc #MCC\_MNC1  }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF\_EN\_OTHER\_ENTERPRISE\_PROF\_UM\_ENT\_CONF | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC1  },  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable},  tagList ‘99BD’H –- Tag for PPR, EnterpriseConfig  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  },  enterpriseConfiguration {  enterpriseOid #S\_ENTERPRISE\_OID,  enterpriseName #ENTERPRISE\_NAME1,  }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF\_ENTERPRISE\_REF\_RULE3 | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC1  },  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable},  taglist 'BD'H  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  },  enterpriseConfiguration {  enterpriseOid #S\_ENTERPRISE\_OID,  enterpriseName #ENTERPRISE\_NAME1,  enterpriseRules {  enterpriseRuleBits {  referenceEnterpriseRule,  onlyEnterpriseProfilesCanBeInstalled  },  numberOfNonEnterpriseProfiles 0  }  }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF\_NO\_ENTERPRISE\_CONF\_TAGLIST\_ENT\_CONF | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC1  },  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable},  tagList ‘BD’H –- Tag for EnterpriseConfig  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF\_REF\_PROF\_ONLY\_THIS\_EN | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC1  },  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable}  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  },  enterpriseConfiguration {  enterpriseOid #S\_ENTERPRISE\_OID,  enterpriseName #ENTERPRISE\_NAME1,  enterpriseRules {  enterpriseRuleBits {  referenceEnterpriseRule,  priorityEnterpriseProfile,  onlyEnterpriseProfilesCanBeInstalled  },  numberOfNonEnterpriseProfiles 0  }  }  } |
| METADATA\_OP\_PROF2\_RPM\_CONF\_ALL\_ENTERP\_RULES | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF2,  serviceProviderName #SP\_NAME2,  profileName #NAME\_OP\_PROF2,  iconType png,  icon #ICON\_OP\_PROF2,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC2  },  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable, disable, delete, listProfileInfo, contactPcmp}  tagList ‘99BA9BBCBD9F1F’H –- Tags for rpmConfiguration, hriServerAddress, lprConfiguration, EnterpriseConfiguration, ServiceDescription  }  },  profileOwnerOid #S\_PROFILE\_OWNER\_OID  },  enterpriseConfiguration {  enterpriseOid #S\_ENTERPRISE\_OID,  enterpriseName #ENTERPRISE\_NAME1,  enterpriseRules {  enterpriseRuleBits {  referenceEnterpriseRule,  priorityEnterpriseProfile,  onlyEnterpriseProfilesCanBeInstalled  },  numberOfNonEnterpriseProfiles 0  }  },  serviceDescription {voice, data}  } |
| METADATA\_OP\_PROF2\_RPM\_CONF\_ALL\_NO\_ENTERP\_CONF | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF2,  serviceProviderName #SP\_NAME2,  profileName #NAME\_OP\_PROF2,  iconType png,  icon #ICON\_OP\_PROF2,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC2  },  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable, disable, delete, listProfileInfo, contactPcmp}  tagList ‘99BA9BBC9F1F’H –- Tags for rpmConfiguration, hriServerAddress, lprConfiguration, ServiceDescription  }  },  profileOwnerOid #S\_PROFILE\_OWNER\_OID  },  serviceDescription {voice, data}  } |
| METADATA\_OP\_PROF2\_RPM\_CONF\_ALLOW\_NON\_ENTERPRISE\_PROF\_INS | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF2,  serviceProviderName #SP\_NAME2,  profileName #NAME\_OP\_PROF2,  iconType png,  icon #ICON\_OP\_PROF2,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC2  },  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable}  }  },  profileOwnerOid #S\_PROFILE\_OWNER\_OID  },  enterpriseConfiguration {  enterpriseOid #S\_ENTERPRISE\_OID,  enterpriseName #ENTERPRISE\_NAME1,  enterpriseRules {  enterpriseRuleBits {  referenceEnterpriseRule,  priorityEnterpriseProfile,  },  numberOfNonEnterpriseProfiles 0  }  }  } |
| METADATA\_OP\_PROF2\_RPM\_CONF\_EN\_ONLY\_THIS\_REF\_ENTERPRISE\_PROF | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF2,  serviceProviderName #SP\_NAME2,  profileName #NAME\_OP\_PROF2,  iconType png,  icon #ICON\_OP\_PROF2,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC2  },  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable}  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  },  enterpriseConfiguration {  enterpriseOid #S\_ENTERPRISE\_OID,  enterpriseName #ENTERPRISE\_NAME1,  enterpriseRules {  enterpriseRuleBits {  referenceEnterpriseRule,  priorityEnterpriseProfile,  onlyEnterpriseProfilesCanBeInstalled,  },  numberOfNonEnterpriseProfiles 0  }  }  } |
| METADATA\_OP\_PROF2\_RPM\_CONF\_EN\_PPR | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF2,  serviceProviderName #SP\_NAME2,  profileName #NAME\_OP\_PROF2,  iconType png,  icon #ICON\_OP\_PROF2,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC2  },  profilePolicyRules {ppr1, ppr2},  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable}  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  } |
| METADATA\_OP\_PROF4 | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF4,  serviceProviderName #SP\_NAME4,  profileName #NAME\_OP\_PROF4,  iconType png,  icon #ICON\_OP\_PROF4,  profileClass operational,  notificationConfigurationInfo {  { profileManagementOperation {  notificationInstall,  notificationLocalEnable,  notificationLocalDisable,  notificationLocalDelete  },  notificationAddress #TEST\_DP\_ADDRESS4  }  },  profileOwner {  mccMnc #MCC\_MNC4  },  profilePolicyRules {  ppr1  }  } |
| METADATA\_OP\_PROF9 | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF9,  serviceProviderName #SP\_NAME9,  profileName #NAME\_OP\_PROF9,  profileOwner {  mccMnc #MCC\_MNC9,  gid1 #GID1,  gid2 #GID2  },  profilePolicyRules {  ppr2  }  } |
| METADATA\_OP\_PROF10 | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF10,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF5,  iconType png,  icon #ICON\_OP\_PROF5,  profileClass operational,  notificationConfigurationInfo {  { profileManagementOperation {  notificationInstall,  notificationLocalEnable,  notificationLocalDisable,  notificationLocalDelete  },  notificationAddress #TEST\_DP\_ADDRESS1  }  },  profileOwner {  mccMnc #MCC\_MNC2  }  } |
| METADATA\_OP\_PROF10\_NO\_PROFILE\_OWNER | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF10,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF5,  iconType png,  icon #ICON\_OP\_PROF5,  profileClass operational,  notificationConfigurationInfo {  { profileManagementOperation {  notificationInstall,  notificationLocalEnable,  notificationLocalDisable,  notificationLocalDelete  },  notificationAddress #TEST\_DP\_ADDRESS1  }  }  } |
| METADATA\_OP1\_GID1GID2\_PRESENT | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  profileOwner {  mccMnc #MCC\_MNC1,  gid1 #GID1,  gid2 #GID2  },  profilePolicyRules {ppr2}  } |
| METADATA\_OP9\_GID1GID2\_MISSING | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF9,  serviceProviderName #SP\_NAME9,  profileName #NAME\_OP\_PROF9,  profileOwner {  mccMnc #MCC\_MNC9  }  } |
| METADATA\_PPR\_NO\_OWNER | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  profilePolicyRules {ppr2}  } |
| METADATA\_SERVICE\_SPECIFIC\_STORED | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  notificationConfigurationInfo {  { profileManagementOperation {  notificationInstall,  notificationLocalEnable,  notificationLocalDisable,  notificationLocalDelete  },  notificationAddress #TEST\_DP\_ADDRESS1  }  },  profileOwner {  mccMnc #MCC\_MNC1  },  serviceSpecificDataStoredInEuicc #VENDOR\_SPECIFIC\_EXTENSION1  } |
| METADATA\_SERVICE\_SPECIFIC\_NOT\_STORED | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  notificationConfigurationInfo {  { profileManagementOperation {  notificationInstall,  notificationLocalEnable,  notificationLocalDisable,  notificationLocalDelete  },  notificationAddress #TEST\_DP\_ADDRESS1  }  },  profileOwner {  mccMnc #MCC\_MNC1  },  serviceSpecificDataNotStoredInEuicc #VENDOR\_SPECIFIC\_EXTENSION2  } |
| METADATA\_SERVICE\_SPECIFIC\_STORED\_AND\_NOT\_STORED | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  notificationConfigurationInfo {  { profileManagementOperation {  notificationInstall,  notificationLocalEnable,  notificationLocalDisable,  notificationLocalDelete  },  notificationAddress #TEST\_DP\_ADDRESS1  }  },  profileOwner {  mccMnc #MCC\_MNC1  },  serviceSpecificDataStoredInEuicc #VENDOR\_SPECIFIC\_EXTENSION1,  serviceSpecificDataNotStoredInEuicc #VENDOR\_SPECIFIC\_EXTENSION2  } |
| METADATA\_WILDCARD | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  profileOwner {  mccMnc #MCC\_MNC\_WILDCARD  },  profilePolicyRules {ppr2}  } |
| METADATA\_WITH\_EC | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  profileOwner {  mccMnc #MCC\_MNC1  },  enterpriseConfiguration {  enterpriseOid #ENTERPRISE\_OID  enterpriseName #ENTERPRISE\_NAME1  }  } |
| METADATA\_WITH\_EC\_OID2 | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF2,  serviceProviderName #SP\_NAME2,  profileName #NAME\_OP\_PROF2,  profileOwner {  mccMnc #MCC\_MNC2  },  enterpriseConfiguration {  enterpriseOid #S\_ENTERPRISE\_OID2  enterpriseName #ENTERPRISE\_NAME1  }  } |
| METADATA\_WITH\_EC\_PROF2 | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF2,  serviceProviderName #SP\_NAME2,  profileName #NAME\_OP\_PROF2,  profileOwner {  mccMnc #MCC\_MNC2  },  enterpriseConfiguration {  enterpriseOid #S\_ENTERPRISE\_OID  enterpriseName #ENTERPRISE\_NAME1  }  } |
| METADATA\_WITH\_ER | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  profileOwner {  mccMnc #MCC\_MNC1  },  enterpriseConfiguration {  enterpriseOid #S\_ENTERPRISE\_OID  enterpriseName #ENTERPRISE\_NAME1  enterpriseRules {  enterpriseRuleBits {  priorityEnterpriseProfile,  onlyEnterpriseProfilesCanBeInstalled  },  numberOfNonEnterpriseProfiles 0  }  }  } |
| METADATA\_WITH\_JPG | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType jpg,  icon #ICON\_JPG  } |
| METADATA\_WITH\_NOTIFS | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  notificationConfigurationInfo {  { profileManagementOperation {  notificationInstall  },  notificationAddress #TEST\_DP\_ADDRESS3  },  { profileManagementOperation {  notificationInstall  },  notificationAddress #TEST\_DP\_ADDRESS2  },  { profileManagementOperation {  notificationLocalEnable  },  notificationAddress #TEST\_DP\_ADDRESS2  },  { profileManagementOperation {  notificationLocalEnable  },  notificationAddress #TEST\_DP\_ADDRESS3  },  { profileManagementOperation {  notificationLocalDisable  },  notificationAddress #TEST\_DP\_ADDRESS3  },  { profileManagementOperation {  notificationLocalDisable  },  notificationAddress #TEST\_DP\_ADDRESS4  },  { profileManagementOperation {  notificationLocalDelete  },  notificationAddress #TEST\_DP\_ADDRESS1  },  { profileManagementOperation {  notificationLocalDelete  },  notificationAddress #TEST\_DP\_ADDRESS3  }  }  } |
| METADATA\_WITH\_PPR1\_PPR2 | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  profileOwner {  mccMnc #MCC\_MNC1  },  profilePolicyRules {ppr1,ppr2}  } |
| METADATA\_WITH\_PPR2 | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  profileOwner {  mccMnc #MCC\_MNC1  },  profilePolicyRules {ppr2}  } |
| METADATA\_WITH\_RER | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  profileOwner {  mccMnc #MCC\_MNC1  },  enterpriseConfiguration {  enterpriseOid #S\_ENTERPRISE\_OID  enterpriseName #ENTERPRISE\_NAME1  enterpriseRules {  enterpriseRuleBits {  referenceEnterpriseRule,  priorityEnterpriseProfile,  onlyEnterpriseProfilesCanBeInstalled  },  numberOfNonEnterpriseProfiles 0  }  }  } |
| METADATA\_WITH\_RER\_PROF2 | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF2,  serviceProviderName #SP\_NAME2,  profileName #NAME\_OP\_PROF2,  profileOwner {  mccMnc #MCC\_MNC2  },  enterpriseConfiguration {  enterpriseOid #ENTERPRISE\_OID  enterpriseName #ENTERPRISE\_NAME1  enterpriseRules {  enterpriseRuleBits {  referenceEnterpriseRule,  onlyEnterpriseProfilesCanBeInstalled  },  numberOfNonEnterpriseProfiles 0  }  }  } |
| METADATA\_WITHOUT\_ICON | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType jpg  } |
| REPLACE\_S\_KEYS\_REQ | req ReplaceSessionKeysRequest ::={  initialMacChainingValue <PPK\_INIT\_MAC>,  ppkEnc <PPK\_ENC>,  ppkCmac <PPK\_MAC>  } |
| REPLACE\_S\_KEYS\_REQ\_INV\_SIZE | req ReplaceSessionKeysRequest ::={  initialMacChainingValue #PPK\_INIT\_MAC\_INV\_SIZE,  ppkEnc #PPK\_ENC\_INV\_SIZE,  ppkCmac #PPK\_MAC\_INV\_SIZE  } |
| S\_INIT\_SC\_PROF1 | req InitialiseSecureChannelRequest ::={  remoteOpId #REMOTE\_OP\_ID\_INSTALL,  transactionId <S\_TRANSACTION\_ID>,  controlRefTemplate {  keyType #KEY\_TYPE,  keyLen #KEY\_LENGTH,  hostId #HOST\_ID  },  smdpOtpk <OTPK\_S\_SM\_DP+\_ECKA>,  smdpSign <S\_SM\_DP+\_SIGN>  } |
| S\_INIT\_SC\_PROF2 | req InitialiseSecureChannelRequest ::={  remoteOpId #REMOTE\_OP\_ID\_INSTALL,  transactionId <S\_TRANSACTION\_ID>,  controlRefTemplate {  keyType #KEY\_TYPE,  keyLen #KEY\_LENGTH,  hostId #HOST\_ID  },  smdpOtpk <OTPK\_S\_SM\_DP+\_ECKA>,  smdpSign <S\_SM\_DP+\_SIGN>  } |

D.2 VOID

D.3 ES10x Requests And Responses

D.3.1 ES10x Requests

| Name | Content |
| --- | --- |
| AUTHENTICATE\_SMDP\_WITH\_DEVICE\_INFO\_NAI | req AuthenticateServerRequest ::= {  serverSigned1 {  transactionId <S\_TRANSACTION\_ID>,  euiccChallenge <EUICC\_CHALLENGE>,  serverAddress #TEST\_DP\_ADDRESS1,  serverChallenge <S\_SMDP\_CHALLENGE>  },  serverSignature1 <S\_SMDP\_SIGNATURE1>,  euiccCiPKIdToBeUsed  <EUICC\_CI\_PK\_ID\_TO\_BE\_USED>,  serverCertificate #CERT\_S\_SM\_DPauth\_ECDSA,  ctxParams1 #CTX\_PARAMS1\_DEVICE\_INFO\_NAI  } |
| CANCEL\_SESSION\_INV\_TRANS\_ID | req CancelSessionRequest ::={  transactionId <INVALID\_TRANSACTION\_ID>,  reason endUserRejection  } |
| CANCEL\_SESSION\_LOAD\_BPP | req CancelSessionRequest ::={  transactionId <S\_TRANSACTION\_ID>,  reason loadBppExecutionError  } |
| CANCEL\_SESSION\_METADATA | req CancelSessionRequest ::={  transactionId <S\_TRANSACTION\_ID>,  reason metadataMismatch  } |
| CANCEL\_SESSION\_POSTPONED | req CancelSessionRequest ::={  transactionId <S\_TRANSACTION\_ID>,  reason postponed  } |
| CANCEL\_SESSION\_PPR | req CancelSessionRequest ::={  transactionId <S\_TRANSACTION\_ID>,  reason pprNotAllowed  } |
| CANCEL\_SESSION\_REJECT | req CancelSessionRequest ::={  transactionId <S\_TRANSACTION\_ID>,  reason endUserRejection  } |
| CANCEL\_SESSION\_TIMEOUT | req CancelSessionRequest ::={  transactionId <S\_TRANSACTION\_ID>,  reason timeout  } |
| CANCEL\_SESSION\_UNDEF | req CancelSessionRequest ::={  transactionId <S\_TRANSACTION\_ID>,  reason undefinedReason  } |
| ENTERPRISE\_CONFIG1 | {  enterpriseOid #S\_ENTERPRISE\_OID,  enterpriseName #ENTERPRISE\_NAME1,  enterpriseRules {  enterpriseRuleBits {  referenceEnterpriseRule,  priorityEnterpriseProfile,  onlyEnterpriseProfilesCanBeInstalled  },  numberOfNonEnterpriseProfiles 0  }  } |
| ENTERPRISE\_CONFIG1\_EC | {  enterpriseOid #S\_ENTERPRISE\_OID,  enterpriseName #ENTERPRISE\_NAME1,  } |
| ENTERPRISE\_CONFIG1\_ER | {  enterpriseOid #S\_ENTERPRISE\_OID,  enterpriseName #ENTERPRISE\_NAME1,  enterpriseRules {  enterpriseRuleBits {  priorityEnterpriseProfile,  onlyEnterpriseProfilesCanBeInstalled  },  numberOfNonEnterpriseProfiles 0  }  } |
| ENTERPRISE\_CONFIG3 | {  enterpriseOid #ENTERPRISE\_OID  enterpriseName #ENTERPRISE\_NAME1  enterpriseRules {  enterpriseRuleBits {  referenceEnterpriseRule,  priorityEnterpriseProfile,  onlyEnterpriseProfilesCanBeInstalled  },  numberOfNonEnterpriseProfiles 0  }  } |
| ENTERPRISE\_CONFIG4 | {  enterpriseOid #ENTERPRISE\_OID  enterpriseName #ENTERPRISE\_NAME1  enterpriseRules {  enterpriseRuleBits {  onlyEnterpriseProfilesCanBeInstalled  },  numberOfNonEnterpriseProfiles 0  }  } |
| ENTERPRISE\_CONFIG5 | {  enterpriseOid #S\_ENTERPRISE\_OID,  enterpriseName #ENTERPRISE\_NAME1,  } |
| EUICC\_MEMORY\_RESET | req EuiccMemoryResetRequest ::= {  resetOptions {  deleteOperationalProfiles,  resetDefaultSmdpAddress  }  } |
| EUICC\_MEMORY\_RESET\_DEF\_SMDPADDRESS | req EuiccMemoryResetRequest ::= {  resetOptions { resetDefaultSmdpAddress }  } |
| EUICC\_MEMORY\_RESET\_OP\_PRO | req EuiccMemoryResetRequest ::= {  resetOptions { deleteOperationalProfiles }  } |
| GET\_CONF\_OP\_PROF1 | opConfProf1Req ProfileInfoListRequest ::= {  searchCriteria iccid: #ICCID\_OP\_PROF1,  tagList '4FB8'H  } |
| GET\_EID | getEIDReq GetEuiccDataRequest ::= {  tagList '5A'H  } |
| GET\_EID\_INVALID | getEIDReq GetEuiccDataRequest ::= {  tagList '6B'H  } |
| GET\_ENTERPRISE\_CONFIG\_OP\_PROF1 | opConfProf1Req ProfileInfoListRequest ::= {  searchCriteria iccid: #ICCID\_OP\_PROF1,  tagList '5ABD'H  } |
| GET\_ENTERPRISE\_CONFIG\_OP\_PROF2 | opConfProf1Req ProfileInfoListRequest ::= {  searchCriteria iccid: #ICCID\_OP\_PROF2,  tagList '5ABD'H  } |
| GET\_EUICC\_CHALLENGE | request GetEuiccChallengeRequest ::= {} |
| GET\_EUICC\_CONFIGURED\_DATA | request EuiccConfiguredDataRequest ::={} |
| GET\_EUICC\_INFO1 | request GetEuiccInfo1Request::= { } |
| GET\_EUICC\_INFO2 | request GetEuiccInfo2Request::= { } |
| GET\_HRI\_SRV\_ADDRESS\_OP\_PROF1 | opConfProf1Req ProfileInfoListRequest ::= {  searchCriteria iccid: #ICCID\_OP\_PROF1,  tagList '5A9B'H  } |
| GET\_LPR\_CONFIG\_OP\_PROF1 | opConfProf1Req ProfileInfoListRequest ::= {  searchCriteria iccid: #ICCID\_OP\_PROF1,  tagList '5ABC'H  } |
| GET\_METADATA\_OP\_PROF1 | opConfProf1Req ProfileInfoListRequest ::= {  searchCriteria iccid: #ICCID\_OP\_PROF1,  tagList '5A9192939495B6B799'H  } |
| GET\_METADATA\_OP\_PROF1\_SERVICE\_SPECIFIC | opConfProf1Req ProfileInfoListRequest ::= {  searchCriteria iccid: #ICCID\_OP\_PROF1,  tagList '5A9192939495B6B799BF22'H  } |
| GET\_MULTIPLE\_TAGS\_OP\_PROF1 | opConfProf1Req ProfileInfoListRequest ::= {  searchCriteria iccid: #ICCID\_OP\_PROF1,  tagList 'BA9BBC'H  } |
| GET\_NEW\_METADATA\_V3 | getupdate1Req ProfileInfoListRequest ::= {  searchCriteria iccid: #ICCID\_OP\_PROF1,  tagList '9192939499B6BF22BA9BBCBF20'H *-- names, icon, PPRs, notif config, specific data, RPM configuration, HRI Server Address, LPR configuration and DC configuration*  } |
| GET\_NEW\_METADATA\_NOTIF | getupdate1Req ProfileInfoListRequest ::= {  searchCriteria iccid: #ICCID\_OP\_PROF1,  tagList '91929394B699'H *-- names, icon, notif config and PPRs*  } |
| GET\_NEW\_METADATA\_SPEC\_DATA | getupdate1Req ProfileInfoListRequest ::= {  searchCriteria iccid: #ICCID\_OP\_PROF1,  tagList '91929394BF2299'H *-- names, icon, specific data and PPRs*  } |
| GET\_NEW\_METADATA\_RPM\_CONFIG | getupdate1Req ProfileInfoListRequest ::= {  searchCriteria iccid: #ICCID\_OP\_PROF1,  tagList '91929394BA99'H *-- names, icon, RPM configuration and PPRs*  } |
| GET\_NEW\_METADATA\_HRI\_ADDR | getupdate1Req ProfileInfoListRequest ::= {  searchCriteria iccid: #ICCID\_OP\_PROF1,  tagList '919293949B99'H *-- names, icon, HRI Server Address and PPRs*  } |
| GET\_NEW\_METADATA\_LPR\_CONFIG | getupdate1Req ProfileInfoListRequest ::= {  searchCriteria iccid: #ICCID\_OP\_PROF1,  tagList '91929394BC99'H *-- names, icon, LPR configuration and PPRs*  } |
| GET\_NEW\_METADATA\_DC\_CONFIG | getupdate1Req ProfileInfoListRequest ::= {  searchCriteria iccid: #ICCID\_OP\_PROF1,  tagList '91929394BF2099'H *-- names, icon, DC configuration and PPRs*  } |
| GET\_NOTIF\_CONF\_OP\_PROF1 | opConfProf1Req ProfileInfoListRequest ::= {  searchCriteria iccid: #ICCID\_OP\_PROF1,  tagList '5AB6'H  } |
| GET\_PPR\_OP\_PROF1 | opConfProf1Req ProfileInfoListRequest ::= {  searchCriteria iccid: #ICCID\_OP\_PROF1,  tagList '5A99'H  } |
| GET\_PROFILES\_INFO\_ALL | request ProfileInfoListRequest::= { } |
| GET\_PROFILES\_INFO\_ICCID\_TAGLIST1 | request ProfileInfoListRequest::= {  searchCriteria iccid: #ICCID\_OP\_PROF1,  tagList '9F70'H *--state* } |
| GET\_PROFILES\_INFO\_ICCID\_TAGLIST2 | request ProfileInfoListRequest::= {  searchCriteria iccid: #ICCID\_OP\_PROF1,  tagList '93'H *--icon type* } |
| GET\_PROFILES\_INFO\_ICCID\_TAGLIST3 | request ProfileInfoListRequest::= {  searchCriteria iccid: #ICCID\_OP\_PROF1,  tagList '95'H *--Profile Class* } |
| GET\_PROFILES\_INFO\_ICCID\_TAGLIST4 | request ProfileInfoListRequest::= {  searchCriteria iccid: #ICCID\_OP\_PROF1,  tagList 'B6'H *--Notification configuration* } |
| GET\_PROFILES\_INFO\_ICCID\_TAGLIST5 | request ProfileInfoListRequest::= {  searchCriteria iccid: #ICCID\_OP\_PROF3,  tagList '99'H *--ppr* } |
| GET\_PROFILES\_INFO\_OPTAGLIST1 | request ProfileInfoListRequest::= {  searchCriteria profileClass: operational,  tagList '5A9F70'H *-- ICCID and State* } |
| GET\_PROFILES\_INFO\_OPTAGLIST2 | request ProfileInfoListRequest::= {  searchCriteria profileClass: operational,  tagList '909F70'H *--Nickname and State* } |
| GET\_PROFILES\_INFO\_OPTAGLIST3 | request ProfileInfoListRequest::= {  searchCriteria profileClass: operational,  tagList '9493'H *--Icon, Icon type* } |
| GET\_PROFILES\_INFO\_OPTAGLIST4 | request ProfileInfoListRequest::= {  searchCriteria profileClass: operational,  tagList '949F70'H *--Icon, state* } |
| GET\_PROFILES\_INFO\_PROFCLASS | request ProfileInfoListRequest::= {  searchCriteria profileClass: operational } |
| GET\_PROFILES\_INFO\_TAGLIST\_ICCID | request ProfileInfoListRequest::= {  tagList '5A'H } |
| GET\_PROFILES\_INFO\_TAGLIST\_ICON | request ProfileInfoListRequest::= {  tagList '94'H } |
| GET\_PROFILES\_INFO\_TAGLIST\_ISDPAID | request ProfileInfoListRequest::= {  tagList '4F'H } |
| GET\_PROFILES\_INFO\_TAGLIST\_PROFILE\_NAME | request ProfileInfoListRequest::= {  tagList '92'H } |
| GET\_PROFILES\_INFO\_TAGLIST\_PROFILE\_NICKNAME | request ProfileInfoListRequest::= {  tagList '90'H } |
| GET\_PROFILES\_INFO\_TAGLIST\_PROFILE\_OWNER | request ProfileInfoListRequest::= {  tagList 'B7'H } |
| GET\_PROFILES\_INFO\_TAGLIST\_SMDP\_PROP\_DATA | request ProfileInfoListRequest::= {  tagList 'B8'H } |
| GET\_PROFILES\_INFO\_TAGLIST\_SP\_NAME | request ProfileInfoListRequest::= {  tagList '91'H } |
| GET\_PROFILES\_INFO\_TAGLIST1 | request ProfileInfoListRequest::= {  tagList '5A9F70'H *-- ICCID and State* } |
| GET\_PROFILES\_INFO\_TAGLIST2 | request ProfileInfoListRequest::= {  tagList '909F70'H *--Nickname and State* } |
| GET\_PROFILES\_INFO\_TAGLIST3 | request ProfileInfoListRequest::= {  tagList '9493'H  *--Icon, Icon type* } |
| GET\_PROFILES\_INFO\_TAGLIST4 | request ProfileInfoListRequest::= {  tagList '949F70'H *--Icon, state* } |
| GET\_PROFILES\_OWNERS | request ProfileInfoListRequest::= {  tagList 'B7'H } |
| GET\_RAT | request GetRatRequest ::={} |
| GET\_RPM\_CONFIG\_OP\_PROF1 | opConfProf1Req ProfileInfoListRequest ::= {  searchCriteria iccid: #ICCID\_OP\_PROF1,  tagList '5ABA'H  } |
| LIST\_NOTIF\_ALL | request ListNotificationRequest ::= {  profileManagementOperation {  notificationInstall,  notificationLocalEnable,  notificationLocalDisable,  notificationLocalDelete  } } |
| LIST\_NOTIF\_ALL\_RPM | request ListNotificationRequest ::= {  profileManagementOperation {  notificationRpmEnable,  notificationRpmDisable,  notificationRpmDelete,  loadRpmPackageResult  } } |
| LIST\_NOTIF\_DELETE | request ListNotificationRequest ::= {  profileManagementOperation {  notificationLocalDelete  } } |
| LIST\_NOTIF\_DELETE\_ENABLE\_RPM | request ListNotificationRequest ::= {  profileManagementOperation {  notificationRpmDelete,  notificationRpmEnable  } } |
| LIST\_NOTIF\_DELETE\_RPM | request ListNotificationRequest ::= {  profileManagementOperation {  notificationRpmDelete  } } |
| LIST\_NOTIF\_DISABLE | request ListNotificationRequest ::= {  profileManagementOperation {  notificationLocalDelete  } } |
| LIST\_NOTIF\_DISABLE\_DELETE | request ListNotificationRequest ::= {  profileManagementOperation {  notificationLocalDisable,  notificationLocalDelete  } } |
| LIST\_NOTIF\_DISABLE\_DELETE\_RPM | request ListNotificationRequest ::= {  profileManagementOperation {  notificationRpmDisable,  notificationRpmDelete  } } |
| LIST\_NOTIF\_DISABLE\_ENABLE | request ListNotificationRequest ::= {  profileManagementOperation {  notificationLocalDisable,  notificationLocalEnable  } } |
| LIST\_NOTIF\_DISABLE\_ENABLE\_RPM | request ListNotificationRequest ::= {  profileManagementOperation {  notificationRpmDisable,  notificationRpmEnable  } } |
| LIST\_NOTIF\_DISABLE\_RPM | request ListNotificationRequest ::= {  profileManagementOperation {  notificationRpmDisable  } } |
| LIST\_NOTIF\_ENABLE | request ListNotificationRequest ::= {  profileManagementOperation {  notificationLocalEnable  } } |
| LIST\_NOTIF\_ENABLE\_DISABLE\_DELETE | request ListNotificationRequest ::= {  profileManagementOperation {  notificationLocalEnable,  notificationLocalDisable,  notificationLocalDelete  } } |
| LIST\_NOTIF\_ENABLE\_RPM | request ListNotificationRequest ::= {  profileManagementOperation {  notificationRpmEnable  } } |
| LIST\_NOTIF\_INSTALL | request ListNotificationRequest ::= {  profileManagementOperation {  notificationInstall  } } |
| LIST\_NOTIF\_INSTALL\_ENABLE | request ListNotificationRequest ::= {  profileManagementOperation {  notificationInstall,  notificationLocalEnable  } } |
| LIST\_NOTIF\_INSTALL\_ENABLE\_DISABLE | request ListNotificationRequest ::= {  profileManagementOperation {  notificationInstall,  notificationLocalEnable,  notificationLocalDisable  } } |
| LIST\_NOTIF\_NONE | request ListNotificationRequest ::= {  profileManagementOperation {} } |
| LIST\_NOTIF\_OMITTED | request ListNotificationRequest ::= {} |
| LIST\_NOTIF\_RPR\_DELETE\_ENABLE\_RPM | request ListNotificationRequest ::= {  profileManagementOperation {  loadRpmPackageResult,  notificationRpmDelete,  notificationRpmEnable  } } |
| LIST\_NOTIF\_RPR\_DELETE\_RPM | request ListNotificationRequest ::= {  profileManagementOperation {  loadRpmPackageResult,  notificationRpmDelete  } } |
| LIST\_NOTIF\_RPR\_DISABLE\_DELETE\_RPM | request ListNotificationRequest ::= {  profileManagementOperation {  loadRpmPackageResult,  notificationRpmDisable,  notificationRpmDelete  } } |
| LIST\_NOTIF\_RPR\_DISABLE\_ENABLE\_RPM | request ListNotificationRequest ::= {  profileManagementOperation {  loadRpmPackageResult,  notificationRpmDisable,  notificationRpmEnable  } } |
| LIST\_NOTIF\_RPR\_DISABLE\_RPM | request ListNotificationRequest ::= {  profileManagementOperation {  loadRpmPackageResult,  notificationRpmDisable  } } |
| LIST\_NOTIF\_RPR\_ENABLE\_RPM | request ListNotificationRequest ::= {  profileManagementOperation {  loadRpmPackageResult,  notificationRpmEnable  } } |
| METADATA\_EN\_DI\_DE\_NOTIFS | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  notificationConfigurationInfo {  { profileManagementOperation {  notificationLocalEnable,  notificationLocalDisable,  notificationLocalDelete  },  notificationAddress #TEST\_DP\_ADDRESS1  },  { profileManagementOperation {  notificationLocalEnable,  notificationLocalDisablee,  notificationLocalDelete  },  notificationAddress #TEST\_DP\_ADDRESS2  }  }  } |
| METADATA\_OP\_PROF1\_INST\_DIFF | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  notificationConfigurationInfo {  { profileManagementOperation {  notificationInstall  },  notificationAddress #TEST\_DP\_ADDRESS2  }  },  profileOwner {  mccMnc #MCC\_MNC1  }  } |
| METADATA\_OP\_PROF1\_NO\_INSTALL | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  notificationConfigurationInfo {  {  profileManagementOperation {  notificationLocalEnable,  notificationLocalDisable,  notificationLocalDelete  },  notificationAddress #TEST\_DP\_ADDRESS1  }  },  profileOwner {  mccMnc #MCC\_MNC1  }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  profileClass operational,  notificationConfigurationInfo {  {  profileManagementOperation {  notificationRpmEnable,  notificationRpmDisable,  notificationRpmDelete  },  notificationAddress #TEST\_DP\_ADDRESS1  }  },  profileOwner {  mccMnc #MCC\_MNC1  },  rpmConfiguration {  managingDpList {  {  managingDP #S\_SM\_DP+\_OID,  rpmType { enable, disable, delete,  listProfileInfo, contactPcmp }  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF\_ALL | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC1  },  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable, disable, delete, listProfileInfo, contactPcmp}  tagList ‘99BA9BBC’H –- Tag for PPR, rpmConfiguration, hriServerAddress, lprConfiguration  }  },  profileOwnerOid #S\_PROFILE\_OWNER\_OID  },  lprConfiguration{ # TEST\_PCMP\_ADDRESS1 }  } |
|  |  |
| METADATA\_OP\_PROF1\_RPM\_CONF\_ALL\_CI\_PKI\_RAND | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC1  },  rpmConfiguration {  managingDpList {  {  managingDP #S\_SM\_DP+\_OID,  rpmType { enable, disable, delete, listProfileInfo, contactPcmp}  tagList ‘99BA9BBC’H –- Tags for PPR, rpmConfiguration, hriServerAddress, lprConfiguration  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  allowedCiPKId <CI\_PKI\_RANDOM>,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF\_ALL\_DP\_OID2 | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC1  },  profilePolicyRules {ppr1},  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID2,  rpmType {enable, disable, delete, listProfileInfo, contactPcmp}  tagList ‘99BA9BBC’H –- Tags for PPR, rpmConfiguration, hriServerAddress, lprConfiguration  }  },  profileOwnerOid #S\_PROFILE\_OWNER\_OID  },  hriServerAddress { #TEST\_HRI\_ADDRESS1 }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF\_ALL\_PPR1 | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC1  },  profilePolicyRules {ppr1},  hriServerAddress { #TEST\_HRI\_ADDRESS1 },  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable, disable, delete, listProfileInfo, contactPcmp}  tagList ‘99BA9BBC’H –- Tag for *PPR,* rpmConfiguration, hriServerAddress, lprConfiguration  }  },  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF\_ALL\_LPR\_CONF\_NOT\_ALLOWED | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC1  },  profilePolicyRules {ppr1},  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable, disable, delete, listProfileInfo, contactPcmp}  tagList ‘99BA9B’H –- Tag for PPR, rpmConfiguration and hriServerAddress  }  },  profileOwnerOid #S\_PROFILE\_OWNER\_OID  },  hriServerAddress { #TEST\_HRI\_ADDRESS1 }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF\_CP\_CI\_PKI\_RAND | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC1  },  rpmConfiguration {  managingDpList {  {  managingDP #S\_SM\_DP+\_OID,  rpmType {contactPcmp},  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  allowedCiPKId <CI\_PKI\_RANDOM>,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  },  lprConfiguration {  pcmpAddress #TEST\_PCMP\_ADDRESS1  }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF\_DE | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC1  },  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {delete}  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF\_DE\_CI\_PKI\_RAND | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC1  },  rpmConfiguration {  managingDpList {  {  managingDP #S\_SM\_DP+\_OID,  rpmType {delete},  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  allowedCiPKId <CI\_PKI\_RANDOM>,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF\_DE\_DP\_OID2 | metadataReq StoreMetadataRequest:: = {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC1  },  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID2,  rpmType {delete}  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF\_DE\_PPR2 | metadataReq StoreMetadataRequest:: = {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC1  },  profilePolicyRules {ppr2},  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {delete}  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF\_DI | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC1  },  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {disable}  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF\_DI\_CI\_PKI\_RAND | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC1  },  rpmConfiguration {  managingDpList {  {  managingDP #S\_SM\_DP+\_OID,  rpmType {disable},  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  allowedCiPKId #<CI\_PKI\_RANDOM>,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF\_DI\_DP\_OID2 | metadataReq StoreMetadataRequest:: = {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC1  },  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID2,  rpmType {disable}  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF\_DI\_PPR1 | metadataReq StoreMetadataRequest:: = {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC1  },  profilePolicyRules {ppr1},  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {disable}  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF\_EN | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC1  },  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable}  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF\_EN\_CI\_PKI\_RAND | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC1  },  rpmConfiguration {  managingDpList {  {  managingDP #S\_SM\_DP+\_OID,  rpmType {enable},  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  allowedCiPKId <CI\_PKI\_RANDOM>,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF\_EN\_DP\_OID2 | metadataReq StoreMetadataRequest:: = {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC1  },  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID2,  rpmType {enable}  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF\_EN\_ONLY\_THIS\_REF\_ENTERPRISE\_PROF | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC1  },  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable}  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  },  enterpriseConfiguration {  enterpriseOid #S\_ENTERPRISE\_OID,  enterpriseName #ENTERPRISE\_NAME1,  enterpriseRules {  enterpriseRuleBits {  referenceEnterpriseRule,  priorityEnterpriseProfile,  onlyEnterpriseProfilesCanBeInstalled  },  numberOfNonEnterpriseProfiles 0  }  }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF\_EN\_OTHER\_ENTERPRISE\_PROF | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC1  },  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable},  tagList 'BD'H  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  },  enterpriseConfiguration {  enterpriseOid #S\_ENTERPRISE\_OID,  enterpriseName #ENTERPRISE\_NAME1,  enterpriseRules {  enterpriseRuleBits {},  numberOfNonEnterpriseProfiles 0  }  }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF\_EN\_REF\_ENTERPRISE\_PROF | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC1  },  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable}  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  },  enterpriseConfiguration {  enterpriseOid #S\_ENTERPRISE\_OID,  enterpriseName #ENTERPRISE\_NAME1,  enterpriseRules {  enterpriseRuleBits {  referenceEnterpriseRule,  priorityEnterpriseProfile,  onlyEnterpriseProfilesCanBeInstalled  },  numberOfNonEnterpriseProfiles 0  }  }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF\_NO\_PCMP\_ADDR | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC1  },  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {contactPcmp}  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF\_OSN\_DP2 | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  profileClass operational,  notificationConfigurationInfo {  {  profileManagementOperation {  notificationRpmEnable,  notificationRpmDisable,  notificationRpmDelete  },  notificationAddress #TEST\_DP\_ADDRESS2  }  },  profileOwner {  mccMnc #MCC\_MNC1  },  rpmConfiguration {  managingDpList {  {  managingDP #S\_SM\_DP+\_OID,  rpmType { enable, disable, delete,  listProfileInfo, contactPcmp }  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF\_PCMP | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC1  },  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {contactPcmp}  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  },  lprConfiguration {  pcmpAddress #TEST\_PCMP\_ADDRESS1  }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF\_UM\_CI\_PKI\_RAND | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC1  },  rpmConfiguration {  managingDpList {  {  managingDP #S\_SM\_DP+\_OID,  tagList ‘99BA9BBC’H –- Tag for PPR, rpmConfiguration, hriServerAddress, lprConfiguration  }  },  allowedCiPKId #<CI\_PKI\_RANDOM>,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF\_UM\_PPR\_CTRL\_BIT | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC1  },  profilePolicyRules { pprUpdateControl,ppr1,ppr2 },  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  tagList ‘99’H –- Tag for PPR  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  } |
| METADATA\_OP\_PROF1\_RPM\_CONF\_UPDATE\_MD\_PPR | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC1  },  profilePolicyRules {ppr1, ppr2},  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  tagList ‘99’H –- Tag for PPR  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  } |
| METADATA\_OP\_PROF2 | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF2,  serviceProviderName #SP\_NAME2,  profileName #NAME\_OP\_PROF2,  iconType png,  icon #ICON\_OP\_PROF2,  profileClass operational,  notificationConfigurationInfo {  { profileManagementOperation {  notificationInstall,  notificationLocalEnable,  notificationLocalDisable,  notificationLocalDelete  },  notificationAddress #TEST\_DP\_ADDRESS2  }  },  profileOwner {  mccMnc #MCC\_MNC2  }  } |
| METADATA\_OP\_PROF2\_NO\_INSTALL | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF2,  serviceProviderName #SP\_NAME2,  profileName #NAME\_OP\_PROF2,  iconType png,  icon #ICON\_OP\_PROF2,  profileClass operational,  notificationConfigurationInfo {  {  profileManagementOperation {  notificationLocalEnable,  notificationLocalDisable,  notificationLocalDelete  },  notificationAddress #TEST\_DP\_ADDRESS2  }  },  profileOwner {  mccMnc #MCC\_MNC2  }  } |
| METADATA\_OP\_PROF2\_RPM\_CONF\_ALL | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF2,  serviceProviderName #SP\_NAME2,  profileName #NAME\_OP\_PROF2,  iconType png,  icon #ICON\_OP\_PROF2,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC2  },  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable, disable, delete, listProfileInfo, contactPcmp}  tagList ‘99BA9BBCBD9F1F’H –- Tags for rpmConfiguration, hriServerAddress, lprConfiguration, EnterpriseConfiguration, ServiceDescription  }  },  profileOwnerOid #S\_PROFILE\_OWNER\_OID  },  enterpriseConfiguration {  enterpriseOid #S\_ENTERPRISE\_OID,  enterpriseName #ENTERPRISE\_NAME1,  enterpriseRules {  enterpriseRuleBits {  referenceEnterpriseRule,  priorityEnterpriseProfile,  onlyEnterpriseProfilesCanBeInstalled  },  numberOfNonEnterpriseProfiles 0  }  },  serviceDescription {voice, data}  } |
| METADATA\_OP\_PROF2\_RPM\_CONF\_ALL\_OWNER2 | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF2,  serviceProviderName #SP\_NAME2,  profileName #NAME\_OP\_PROF2,  iconType png,  icon #ICON\_OP\_PROF2,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC2  },  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable, disable, delete, listProfileInfo, contactPcmp}  tagList ‘99BA9BBC9F1F’H –- Tags for PPR, rpmConfiguration, hriServerAddress, lprConfiguration, ServiceDescription  }  },  profileOwnerOid #S\_PROFILE\_OWNER\_OID2  },  serviceDescription {voice, data}  } |
| METADATA\_OP\_PROF2\_RPM\_CONF\_EN | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF2,  serviceProviderName #SP\_NAME2,  profileName #NAME\_OP\_PROF2,  iconType png,  icon #ICON\_OP\_PROF2,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC2  },  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable}  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  } |
| METADATA\_OP\_PROF2\_RPM\_CONF\_EN\_OWNER\_OID1 | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF2,  serviceProviderName #SP\_NAME2,  profileName #NAME\_OP\_PROF2,  iconType png,  icon #ICON\_OP\_PROF2,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC2  },  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable}  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  } |
| METADATA\_OP\_PROF2\_RPM\_CONF\_EN\_OWNER\_OID1\_PPR1 | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF2,  serviceProviderName #SP\_NAME2,  profileName #NAME\_OP\_PROF2,  iconType png,  icon #ICON\_OP\_PROF2,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC2  },  profilePolicyRules {ppr1},  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable}  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  } |
| METADATA\_OP\_PROF2\_RPM\_CONF\_EN\_OWNER\_OID2 | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF2,  serviceProviderName #SP\_NAME2,  profileName #NAME\_OP\_PROF2,  iconType png,  icon #ICON\_OP\_PROF2,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC2  },  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable}  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  profileOwnerOid #S\_PROFILE\_OWNER\_OID2  }  } |
| METADATA\_OP\_PROF2\_RPM\_CONF\_EN\_REF\_ENTERPRISE\_PROF | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF2,  serviceProviderName #SP\_NAME2,  profileName #NAME\_OP\_PROF2,  iconType png,  icon #ICON\_OP\_PROF2,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC2  },  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable}  }  },  pollingAddress #TEST\_DP\_ADDRESS1,  profileOwnerOid #S\_PROFILE\_OWNER\_OID  },  enterpriseConfiguration {  enterpriseOid #S\_ENTERPRISE\_OID,  enterpriseName #ENTERPRISE\_NAME1,  enterpriseRules {  enterpriseRuleBits {  referenceEnterpriseRule,  priorityEnterpriseProfile,  onlyEnterpriseProfilesCanBeInstalled  },  numberOfNonEnterpriseProfiles 0  }  }  } |
| METADATA\_OP\_PROF3 | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF3,  serviceProviderName #SP\_NAME3,  profileName #NAME\_OP\_PROF3,  iconType png,  icon #ICON\_OP\_PROF3,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC2  },  profilePolicyRules { ppr2 }  } |
| METADATA\_OP\_PROF3\_RPM\_CONF\_ALL | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF3,  serviceProviderName #SP\_NAME3,  profileName #NAME\_OP\_PROF3,  iconType png,  icon #ICON\_OP\_PROF3,  profileClass operational,  profileOwner {  mccMnc #MCC\_MNC3  },  hriServerAddress { #TEST\_HRI\_ADDRESS3 },  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable, disable, delete, listProfileInfo}  tagList ‘BA9BBC’H –- Tags forrpmConfiguration, hriServerAddress, lprConfiguration  }  },  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  lprConfiguration {  pcmpAddress #TEST\_PCMP\_ADDRESS3  }  } |
| MULTIPLE\_TAGS\_OP\_PROF1 | rpmConfiguration #RPM\_CONFIG\_OP\_PROF1,  lprConfiguration { pcmpAddress #TEST\_PCMP\_ADDRESS1 },  hriServerAddress #TEST\_HRI\_ADDRESS3 |
| PREP\_DOWN\_INV\_CURVE | req PrepareDownloadRequest ::= {  smdpSigned2 {  transactionId <S\_TRANSACTION\_ID>,  ccRequiredFlag FALSE  },  smdpSignature2 <RANDOM\_SM\_DP+\_SIGN>,  smdpCertificate #CERT\_S\_SM\_DPpb\_INV\_CURVE  } |
| PREP\_DOWNLOAD\_CERT\_SMDP2 | req PrepareDownloadRequest ::= {  smdpSigned2 {  transactionId <S\_TRANSACTION\_ID>,  ccRequiredFlag FALSE  },  smdpSignature2 <S\_SM\_DP+\_SIGNATURE2>,  smdpCertificate #CERT\_S\_SM\_DP2pb\_SIG  } |
| PREP\_DOWNLOAD\_INV\_CERT | req PrepareDownloadRequest ::= {  smdpSigned2 {  transactionId <S\_TRANSACTION\_ID>,  ccRequiredFlag FALSE  },  smdpSignature2 <S\_SM\_DP+\_SIGNATURE2>,  smdpCertificate #CERT\_S\_SM\_DPpb\_INV\_SIGN  } |
| PREP\_DOWNLOAD\_INV\_OID | req PrepareDownloadRequest ::= {  smdpSigned2 {  transactionId <S\_TRANSACTION\_ID>,  ccRequiredFlag FALSE  },  smdpSignature2 <S\_SM\_DP+\_SIGNATURE2>,  smdpCertificate #CERT\_S\_SM\_DPauth\_SIG  } |
| PREP\_DOWNLOAD\_INV\_SIGN | req PrepareDownloadRequest ::= {  smdpSigned2 {  transactionId <S\_TRANSACTION\_ID>,  ccRequiredFlag FALSE  },  smdpSignature2 <S\_SM\_DP+\_SIGNATURE2>,  smdpCertificate #CERT\_S\_SM\_DPpb\_SIG  }  NOTE: The <S\_SM\_DP+\_SIGNATURE2> SHALL NOT be computed using the #SK\_S\_SM\_DPpb\_SIG but SHALL have the same length as for a valid signature. |
| PREP\_DOWNLOAD\_INV\_TRANS\_ID | req PrepareDownloadRequest ::= {  smdpSigned2 {  transactionId <INVALID\_TRANSACTION\_ID>,  ccRequiredFlag FALSE  },  smdpSignature2 <S\_SM\_DP+\_SIGNATURE2>,  smdpCertificate #CERT\_S\_SM\_DPpb\_SIG  } |
| PREP\_DOWNLOAD\_INVALID\_CC | req PrepareDownloadRequest ::= {  smdpSigned2 {  transactionId <S\_TRANSACTION\_ID>,  ccRequiredFlag TRUE  },  smdpSignature2 <S\_SM\_DP+\_SIGNATURE2>,  smdpCertificate #CERT\_S\_SM\_DPpb\_SIG  } |
| PREP\_DOWNLOAD\_NO\_AUTH | req PrepareDownloadRequest ::= {  smdpSigned2 {  transactionId <S\_TRANSACTION\_ID>,  ccRequiredFlag FALSE  },  smdpSignature2 <RANDOM\_SM\_DP+\_SIGN>,  smdpCertificate #CERT\_S\_SM\_DPpb\_SIG  } |
| PREP\_DOWNLOAD\_NO\_CC | req PrepareDownloadRequest ::= {  smdpSigned2 {  transactionId <S\_TRANSACTION\_ID>,  ccRequiredFlag FALSE  },  smdpSignature2 <S\_SM\_DP+\_SIGNATURE2>,  smdpCertificate #CERT\_S\_SM\_DPpb\_SIG  } |
| PREP\_DOWNLOAD\_RETRY\_CC | req PrepareDownloadRequest ::= {  smdpSigned2 {  transactionId <S\_TRANSACTION\_ID>,  ccRequiredFlag TRUE,  bppEuiccOtpk <OTPK\_EUICC\_ECKA>  },  smdpSignature2 <S\_SM\_DP+\_SIGNATURE2>,  hashCc <S\_HASHED\_CC>,  smdpCertificate #CERT\_S\_SM\_DPpb\_SIG  } |
| PREP\_DOWNLOAD\_WITH\_CC | req PrepareDownloadRequest ::= {  smdpSigned2 {  transactionId <S\_TRANSACTION\_ID>,  ccRequiredFlag TRUE  },  smdpSignature2 <S\_SM\_DP+\_SIGNATURE2>,  hashCc <S\_HASHED\_CC>,  smdpCertificate #CERT\_S\_SM\_DPpb\_SIG  } |
| RETRIEVE\_NOTIF\_ALL | request RetrieveNotificationsListRequest ::= {  searchCriteria profileManagementOperation {  notificationInstall,  notificationLocalEnable,  notificationLocalDisable,  notificationLocalDelete  } } |
| RETRIEVE\_NOTIF\_DELETE | request RetrieveNotificationsListRequest::= {  searchCriteria profileManagementOperation {  notificationLocalDelete  } } |
| RETRIEVE\_NOTIF\_DISABLE | request RetrieveNotificationsListRequest::= {  searchCriteria profileManagementOperation {  notificationLocalDisable  } } |
| RETRIEVE\_NOTIF\_DISABLE\_DELETE | request RetrieveNotificationsListRequest ::= {  searchCriteria profileManagementOperation {  notificationLocalDisable,  notificationLocalDelete  } } |
| RETRIEVE\_NOTIF\_DISABLE\_ENABLE | request RetrieveNotificationsListRequest ::= {  searchCriteria profileManagementOperation {  notificationLocalDisable,  notificationLocalEnable  } } |
| RETRIEVE\_NOTIF\_ENABLE | request RetrieveNotificationsListRequest::= {  searchCriteria profileManagementOperation {  notificationLocalEnable  } } |
| RETRIEVE\_NOTIF\_INSTALL | request RetrieveNotificationsListRequest::= {  searchCriteria profileManagementOperation {  notificationInstall  } } |
| RETRIEVE\_NOTIF\_INSTALL\_ENABLE | request RetrieveNotificationsListRequest ::= {  searchCriteria profileManagementOperation {  notificationInstall,  notificationLocalEnable  } } |
| RETRIEVE\_NOTIF\_INSTALL\_ENABLE\_DISABLE | request RetrieveNotificationsListRequest ::= {  searchCriteria profileManagementOperation {  notificationInstall,  notificationLocalEnable,  notificationLocalDisable  } } |
| RETRIEVE\_NOTIF\_NONE | request RetrieveNotificationsListRequest ::= {  searchCriteria profileManagementOperation {} } |
| RETRIEVE\_NOTIF\_OMITTED | request RetrieveNotificationsListRequest ::= {  } |
| RPM\_CONFIG\_OID2 | {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType { enable, disable, delete, listProfileInfo }  tagList ‘99BA9BBC’H  }  },  profileOwnerOid #S\_PROFILE\_OWNER\_OID2  } |
| RPM\_CONFIG\_OP\_PROF1 | {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable, disable, delete, listProfileInfo}  tagList ‘99BA9BBC’H  }  },  profileOwnerOid #S\_PROFILE\_OWNER\_OID  } |
| SET\_EUICC\_CONFIGURED\_ADDRESS\_1 | request SetDefaultDpAddressRequest::={  defaultDpAddress #TEST\_DP\_ADDRESS1  } |
| SET\_EUICC\_CONFIGURED\_ADDRESS\_2 | request SetDefaultDpAddressRequest::={  defaultDpAddress #TEST\_DP\_ADDRESS2  } |
| SET\_EUICC\_CONFIGURED\_ADDRESS\_EMPTY | request SetDefaultDpAddressRequest::={  defaultDpAddress ""  } |
| SET\_NICKNAME\_EMPTY\_OP\_PROF1 | setNicknameReq SetNicknameRequest ::= {  iccid #ICCID\_OP\_PROF1,  profileNickname ""  } |
| SET\_NICKNAME\_ICCID\_UNKNOWN | setNicknameReq SetNicknameRequest ::= {  iccid #ICCID\_UNKNOWN,  profileNickname #NICKNAME2  } |
| SET\_NICKNAME\_OP\_PROF1 | setNicknameReq SetNicknameRequest ::= {  iccid #ICCID\_OP\_PROF1,  profileNickname #NICKNAME2  } |

D.3.2 ES10x Responses

| **Name** | **Content** |
| --- | --- |
| ENTERPRISE\_CONFIG\_NO\_ENT\_RULES | {  enterpriseOid #S\_ENTERPRISE\_OID,  enterpriseName #ENTERPRISE\_NAME1,  } |
| ENTERPRISE\_CONFIG2 | {  enterpriseOid #S\_ENTERPRISE\_OID,  enterpriseName #ENTERPRISE\_NAME1,  enterpriseRules {  enterpriseRuleBits {  referenceEnterpriseRule,  priorityEnterpriseProfile,  onlyEnterpriseProfilesCanBeInstalled  },  numberOfNonEnterpriseProfiles 0  }  } |
| NOTIF\_METADATA\_DELETE1  (NotificationMetadata) | {  seqNumber <NOTIF\_SEQ\_NO\_DE1>,  profileManagementOperation { notificationLocalDelete},  notificationAddress #TEST\_DP\_ADDRESS1,  iccid #ICCID\_OP\_PROF1  } |
| NOTIF\_METADATA\_DELETE2  (NotificationMetadata) | {  seqNumber <NOTIF\_SEQ\_NO\_DE2>,  profileManagementOperation { notificationLocalDelete},  notificationAddress #TEST\_DP\_ADDRESS2,  iccid #ICCID\_OP\_PROF2  } |
| NOTIF\_METADATA\_DISABLE1  (NotificationMetadata) | {  seqNumber <NOTIF\_SEQ\_NO\_DI1>,  profileManagementOperation { notificationLocalDisable},  notificationAddress #TEST\_DP\_ADDRESS1,  iccid #ICCID\_OP\_PROF1  } |
| NOTIF\_METADATA\_DP2\_EN1\_RPM  (NotificationMetadata) | {  seqNumber <NOTIF\_SEQ\_NO\_EN1\_RPM>,  profileManagementOperation { notificationRpmEnable},  notificationAddress #TEST\_DP\_ADDRESS2,  iccid #ICCID\_OP\_PROF1  } |
| NOTIF\_METADATA\_EN1\_RPM  (NotificationMetadata) | {  seqNumber <NOTIF\_SEQ\_NO\_EN1\_RPM>,  profileManagementOperation { notificationRpmEnable },  notificationAddress #TEST\_DP\_ADDRESS1,  iccid #ICCID\_OP\_PROF1  } |
| NOTIF\_METADATA\_ENABLE1  (NotificationMetadata) | {  seqNumber <NOTIF\_SEQ\_NO\_EN1>,  profileManagementOperation { notificationLocalEnable},  notificationAddress #TEST\_DP\_ADDRESS1,  iccid #ICCID\_OP\_PROF1  } |
| NOTIF\_METADATA\_ENABLE2  (NotificationMetadata) | {  seqNumber <NOTIF\_SEQ\_NO\_EN2>,  profileManagementOperation { notificationLocalEnable},  notificationAddress #TEST\_DP\_ADDRESS2,  iccid #ICCID\_OP\_PROF2  } |
| NOTIF\_METADATA\_INSTALL1  (NotificationMetadata) | {  seqNumber <NOTIF\_SEQ\_NO\_IN1>,  profileManagementOperation { notificationInstall },  notificationAddress #TEST\_DP\_ADDRESS1,  iccid #ICCID\_OP\_PROF1  } |
| NOTIF\_METADATA\_INSTALL1\_DP1\_PIR  (NotificationMetadata) | {  seqNumber <NOTIF\_SEQ\_NO\_IN1\_DP1>,  profileManagementOperation { notificationInstall },  notificationAddress #TEST\_DP\_ADDRESS1,  iccid #ICCID\_OP\_PROF1  } |
| NOTIF\_METADATA\_INSTALL1\_DP2\_OSN  (NotificationMetadata) | {  seqNumber <NOTIF\_SEQ\_NO\_IN1\_DP2>,  profileManagementOperation { notificationInstall },  notificationAddress #TEST\_DP\_ADDRESS2,  iccid #ICCID\_OP\_PROF1  } |
| NOTIF\_METADATA\_INSTALL1\_PIR  (NotificationMetadata) | {  seqNumber <NOTIF\_SEQ\_NO\_IN1\_PIR>,  profileManagementOperation {  notificationInstall  },  notificationAddress #TEST\_DP\_ADDRESS1,  iccid #ICCID\_OP\_PROF1  } |
| NOTIF\_METADATA\_INSTALL2  (NotificationMetadata) | {  seqNumber <NOTIF\_SEQ\_NO\_IN2>,  profileManagementOperation {  notificationInstall  },  notificationAddress #TEST\_DP\_ADDRESS2,  iccid #ICCID\_OP\_PROF2  } |
| NOTIF\_METADATA\_INSTALL2\_PIR  (NotificationMetadata) | {  seqNumber <NOTIF\_SEQ\_NO\_IN2\_PIR>,  profileManagementOperation {  notificationInstall  },  notificationAddress #TEST\_DP\_ADDRESS2,  iccid #ICCID\_OP\_PROF2  } |
| NOTIF\_METADATA\_PROF1\_DP1\_RPR  (NotificationMetadata) | {  seqNumber <NOTIF\_SEQ\_NO\_PROF1\_RPR>,  profileManagementOperation { loadRpmPackageResult },  notificationAddress #TEST\_DP\_ADDRESS1  } |
| NOTIF\_METADATA\_RPR  (NotificationMetadata) | {  seqNumber <NOTIF\_SEQ\_NO\_PROF1\_RPR>,  profileManagementOperation { loadRpmPackageResult },  notificationAddress #TEST\_DP\_ADDRESS1,  } |
| NOTIF\_METADATA2\_DELETE1  (NotificationMetadata) | {  seqNumber <NOTIF\_SEQ\_NO2\_DE1>,  profileManagementOperation { notificationLocalDelete},  notificationAddress #TEST\_DP\_ADDRESS2,  iccid #ICCID\_OP\_PROF1  } |
| NOTIF\_METADATA2\_DISABLE1  (NotificationMetadata) | {  seqNumber <NOTIF\_SEQ\_NO2\_DI1>,  profileManagementOperation { notificationLocalDisable},  notificationAddress #TEST\_DP\_ADDRESS2,  iccid #ICCID\_OP\_PROF1  } |
| NOTIF\_METADATA2\_ENABLE1  (NotificationMetadata) | {  seqNumber <NOTIF\_SEQ\_NO2\_EN1>,  profileManagementOperation { notificationLocalEnable},  notificationAddress #TEST\_DP\_ADDRESS2,  iccid #ICCID\_OP\_PROF1  } |
| PPR1\_WITH\_OWNER\_GID  (ProfilePolicyAuthorisationRule) | {  pprIds { ppr1 },  allowedOperators {  { mccMnc #MCC\_MNC2,  gid1 #GID1,  gid2 #GID2  }  },  pprFlags {consentRequired}  } |
| PPR1\_WITHOUT\_GID  (ProfilePolicyAuthorisationRule) | {  pprIds { ppr1 },  allowedOperators {  { mccMnc #MCC\_MNC4 }  },  pprFlags {consentRequired}  } |
| PPR2\_WITHOUT\_CONSENT  (ProfilePolicyAuthorisationRule) | {  pprIds { ppr2 },  allowedOperators {  { mccMnc '92EEEE'H, gid1 ''H, gid2 ''H}  },  pprFlags { }  } |
| PPRS\_ALLOWED  (ProfilePolicyAuthorisationRule) | {  pprIds { ppr1, ppr2 },  allowedOperators {  { mccMnc 'EEEEEE'H, gid1 ''H, gid2 ''H}  },  pprFlags {consentRequired}  } |
| PPR2\_ALLOWED  (ProfilePolicyAuthorisationRule) | {  pprIds { ppr2 },  allowedOperators {  { mccMnc 'EEEEEE'H, gid1 ''H, gid2 ''H}  },  pprFlags {consentRequired}  } |
| PROFILE\_INFO1  (ProfileInfo) | {  iccid #ICCID\_OP\_PROF1,  isdpAid <ISD\_P\_AID1>,  profileState enabled,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational  } |
| PROFILE\_INFO1\_DISABLED  (ProfileInfo) | {  iccid #ICCID\_OP\_PROF1,  isdpAid <ISD\_P\_AID1>,  profileState disabled,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational  } |
| PROFILE\_INFO1\_MEP  (ProfileInfo) | {  iccid #ICCID\_OP\_PROF1,  isdpAid <ISD\_P\_AID1>,  profileState enabled or disabled,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  enabledOnEsimPort <ANY\_PORT\_VALUE>  } |
| PROFILE\_INFO1\_MEPB  (ProfileInfo) | {  iccid #ICCID\_OP\_PROF1,  isdpAid <ISD\_P\_AID1>,  profileState enabled OR disabled,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational  } |
| PROFILE\_INFO2  (ProfileInfo) | {  iccid #ICCID\_OP\_PROF2,  isdpAid <ISD\_P\_AID2>,  profileState disabled,  serviceProviderName #SP\_NAME2,  profileName #NAME\_OP\_PROF2,  iconType png,  icon #ICON\_OP\_PROF2,  profileClass operational  } |
| PROFILE\_INFO2\_ENABLED  (ProfileInfo) | {  iccid #ICCID\_OP\_PROF2,  isdpAid <ISD\_P\_AID2>,  profileState enabled,  serviceProviderName #SP\_NAME2,  profileName #NAME\_OP\_PROF2,  iconType png,  icon #ICON\_OP\_PROF2,  profileClass operational  } |
| PROFILE\_INFO2\_ENABLED\_MEP  (ProfileInfo) | {  iccid #ICCID\_OP\_PROF2,  isdpAid <ISD\_P\_AID2>,  profileState enabled,  serviceProviderName #SP\_NAME2,  profileName #NAME\_OP\_PROF2,  iconType png,  icon #ICON\_OP\_PROF2,  profileClass operational,  enabledOnEsimPort <ANY\_PORT\_VALUE>  } |
| PROFILE\_INFO2\_ENABLED\_ON\_OTHER\_PORT | {  iccid #ICCID\_OP\_PROF2,  isdpAid <ISD\_P\_AID2>,  profileState enabled or disabled,  serviceProviderName #SP\_NAME2,  profileName #NAME\_OP\_PROF2,  iconType png,  icon #ICON\_OP\_PROF2,  profileClass operational,  enabledOnEsimPort <ANY\_PORT\_VALUE>  } |
| PROFILE\_INFO2\_MEPB  (ProfileInfo) | {  iccid #ICCID\_OP\_PROF2,  isdpAid <ISD\_P\_AID2>,  profileState enabled OR disabled,  serviceProviderName #SP\_NAME2,  profileName #NAME\_OP\_PROF2,  iconType png,  icon #ICON\_OP\_PROF2,  profileClass operational  } |
| PROFILE\_INFO3  (ProfileInfo) | {  iccid #ICCID\_OP\_PROF3,  isdpAid <ISD\_P\_AID3>,  profileState disabled,  profileNickname #NICKNAME3,  serviceProviderName #SP\_NAME3,  profileName #NAME\_OP\_PROF3,  iconType png,  icon #ICON\_OP\_PROF3,  profileClass operational  } |
| PROFILE\_INFO3\_ENABLED  (ProfileInfo) | {  iccid #ICCID\_OP\_PROF3,  isdpAid <ISD\_P\_AID3>,  profileState enabled,  profileNickname #NICKNAME3,  serviceProviderName #SP\_NAME3,  profileName #NAME\_OP\_PROF3,  iconType png,  icon #ICON\_OP\_PROF3,  profileClass operational  } |
| PROFILE\_INFO3\_ENABLED\_ON\_OTHER\_PORT  (ProfileInfo) | {  iccid #ICCID\_OP\_PROF3,  isdpAid <ISD\_P\_AID3>,  profileState enabled or disabled,  profileNickname #NICKNAME3,  serviceProviderName #SP\_NAME3,  profileName #NAME\_OP\_PROF3,  iconType png,  icon #ICON\_OP\_PROF3,  profileClass operational,  enabledOnEsimPort <ANY\_PORT\_VALUE>  } |
| PROFILE\_INFO3\_MEPB  (ProfileInfo) | {  iccid #ICCID\_OP\_PROF3,  isdpAid <ISD\_P\_AID3>,  profileState enabled OR disabled,  profileNickname #NICKNAME3,  serviceProviderName #SP\_NAME3,  profileName #NAME\_OP\_PROF3,  iconType png,  icon #ICON\_OP\_PROF3,  profileClass operational  } |
| PROFILE\_INFO4\_ENABLED  (ProfileInfo) | {  iccid #ICCID\_OP\_PROF4,  isdpAid <ISD\_P\_AID4>,  profileState enabled,  serviceProviderName #SP\_NAME4,  profileName #NAME\_OP\_PROF4,  iconType png,  icon #ICON\_OP\_PROF4,  profileClass operational  } |
| PROFILES\_INFO\_ICCID\_TAGLIST1  (ProfileInfo) | {profileState enabled} |
| PROFILES\_INFO\_ICCID\_TAGLIST2  (ProfileInfo) | {iconType png} |
| PROFILES\_INFO\_ICCID\_TAGLIST3  (ProfileInfo) | {profileClass operational } |
| PROFILES\_INFO\_ICCID\_TAGLIST4  (ProfileInfo) | notificationConfigurationInfo from #METADATA\_OP\_PROF1 |
| PROFILES\_INFO\_ICCID\_TAGLIST5  (ProfileInfo) | profilePolicyRules from #METADATA\_OP\_PROF3 |
| PROFILES\_INFO\_ICCID\_TAGLIST6  (ProfileInfo) | profilePolicyRules from METADATA\_OP\_PROF1\_RPM\_CONF\_ALL\_PPR1 |
| PROFILES\_INFO\_RPM\_TAGLIST1  (ProfileInfo) | {  iccid #ICCID\_OP\_PROF1,  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable, disable, delete, listProfileInfo, contactPcmp}  tagList ‘99BA9BBC’H –- Tags for PPR,rpmConfig,hriConfig,lprConfig  }  },  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  } |
| PROFILES\_INFO\_RPM\_TAGLIST2  (ProfileInfo) | {  iccid #ICCID\_OP\_PROF3,  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable, disable, delete, listProfileInfo}  tagList ‘BA9BBC’H –- Tags for PPR,rpmConfig,hriConfig,lprConfig  }  },  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  } |
| PROFILES\_INFO\_RPM\_TAGLIST3  (ProfileInfo) | {  iccid #ICCID\_OP\_PROF1,  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable, disable, delete, listProfileInfo, contactPcmp}  tagList ‘99BA9BBC’H –- Tags for PPR,rpmConfig,hriConfig,lprConfig  }  },  profileOwnerOid #S\_PROFILE\_OWNER\_OID  },  hriServerAddress { #TEST\_HRI\_ADDRESS1 }  } |
| PROFILES\_INFO\_RPM\_TAGLIST4  (ProfileInfo) | {  iccid #ICCID\_OP\_PROF3,  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable, disable, delete, listProfileInfo}  tagList ‘BA9BBC’H –- Tags for rpmConfig,hriConfig,lprConfig  }  },  profileOwnerOid #S\_PROFILE\_OWNER\_OID  },  hriServerAddress { #TEST\_HRI\_ADDRESS3 },  lprConfiguration {  pcmpAddress #TEST\_PCMP\_ADDRESS3  }  } |
| PROFILES\_INFO\_RPM\_TAGLIST5  (ProfileInfo) | {  iccid #ICCID\_OP\_PROF2,  rpmConfiguration {  managingDpList {  { managingDP #S\_SM\_DP+\_OID,  rpmType {enable, disable, delete, listProfileInfo, contactPcmp}  tagList ‘99BA9BBC9F1F’H *–- Tags for PPR,* rpmConfiguration, hriServerAddress, lprConfiguration, ICCID, ServiceDescription  }  },  profileOwnerOid #S\_PROFILE\_OWNER\_OID2  },  serviceDescription {voice, data}  } |
| PROFILES\_INFO\_RPM\_TAGLIST6  (ProfileInfo) | {  iccid #ICCID\_OP\_PROF2,  enterpriseConfiguration {  enterpriseOid #S\_ENTERPRISE\_OID,  enterpriseName #ENTERPRISE\_NAME1,  enterpriseRules {  enterpriseRuleBits {  referenceEnterpriseRule,  priorityEnterpriseProfile,  onlyEnterpriseProfilesCanBeInstalled  },  numberOfNonEnterpriseProfiles 0  }  },  serviceDescription {voice, data}  } |
| PROFILES\_INFO\_TAGLIST\_ICCID  (ProfileInfo) | {iccid #ICCID\_OP\_PROF1},  {iccid #ICCID\_OP\_PROF2},  {iccid #ICCID\_OP\_PROF3} |
| PROFILES\_INFO\_TAGLIST\_ICON  (ProfileInfo) | {icon #ICON\_OP\_PROF1},  {icon #ICON\_OP\_PROF2},  {icon #ICON\_OP\_PROF3} |
| PROFILES\_INFO\_TAGLIST\_ISDPAID  (ProfileInfo) | {isdpAid <ISD\_P\_AID1>},  {isdpAid <ISD\_P\_AID2>},  {isdpAid <ISD\_P\_AID3>} |
| PROFILES\_INFO\_TAGLIST\_PROFILE\_NAME  (ProfileInfo) | {profileName #NAME\_OP\_PROF1},  {profileName #NAME\_OP\_PROF2},  {profileName #NAME\_OP\_PROF3} |
| PROFILES\_INFO\_TAGLIST\_PROFILE\_NICKNAME  (ProfileInfo) | {profileNickname #NICKNAME3} |
| PROFILES\_INFO\_TAGLIST\_PROFILE\_OWNER  (ProfileInfo) | {profileOwner #OWNER\_OP\_PROF1},  {profileOwner #OWNER\_OP\_PROF2},  {profileOwner #OWNER\_OP\_PROF2} |
| PROFILES\_INFO\_TAGLIST\_SMDP\_PROP\_DATA  (ProfileInfo) | {dpProprietaryData #SMDP\_PROP\_DATA1} |
| PROFILES\_INFO\_TAGLIST\_SP\_NAME  (ProfileInfo) | {serviceProviderName #SP\_NAME1},  {serviceProviderName #SP\_NAME2},  {serviceProviderName #SP\_NAME3} |
| PROFILES\_INFO\_TAGLIST1  (ProfileInfo) | {  iccid #ICCID\_OP\_PROF1,  profileState enabled  },  {  iccid #ICCID\_OP\_PROF2,  profileState disabled  },  {  iccid #ICCID\_OP\_PROF3,  profileState disabled  } |
| PROFILES\_INFO\_TAGLIST2  (ProfileInfo) | {  profileState enabled  },  {  profileState disabled  },  {  profileState disabled,  profileNickname #NICKNAME3 } |
| PROFILES\_INFO\_TAGLIST3  (ProfileInfo) | {  iconType png,  icon #ICON\_OP\_PROF1  },  {  iconType png,  icon #ICON\_OP\_PROF2  },  {  iconType png,  icon #ICON\_OP\_PROF3  } |
| PROFILES\_INFO\_TAGLIST4  (ProfileInfo) | {  profileState enabled,  icon #ICON\_OP\_PROF1  },  {  profileState disabled,  icon #ICON\_OP\_PROF2  },  {  profileState disabled,  icon #ICON\_OP\_PROF3  } |
| PROFILES\_INFO\_TAGLIST5\_MEPA1  (ProfileInfo) | {  iccid #ICCID\_OP\_PROF1,   profileState enabled,  enabledOnEsimPort 0x01  },  {  iccid #ICCID\_OP\_PROF2,   profileState enabled,  enabledOnEsimPort 0x02  },  {  iccid #ICCID\_OP\_PROF3,   profileState disabled  } |
| PROFILES\_INFO\_TAGLIST5\_MEPA2\_12  (ProfileInfo) | {  iccid #ICCID\_OP\_PROF1,  profileState enabled,  enabledOnEsimPort 0x01  },  {  iccid #ICCID\_OP\_PROF2,  profileState enabled,  enabledOnEsimPort 0x02  },  {  iccid #ICCID\_OP\_PROF3,  profileState disabled  } |
| PROFILES\_INFO\_TAGLIST5\_MEPA2\_21  (ProfileInfo) | {  iccid #ICCID\_OP\_PROF1,  profileState enabled,  enabledOnEsimPort 0x02  },  {  iccid #ICCID\_OP\_PROF2,  profileState enabled,  enabledOnEsimPort 0x01  },  {  iccid #ICCID\_OP\_PROF3,  profileState disabled  } |
| PROFILES\_INFO\_TAGLIST5\_MEPB  (ProfileInfo) | {  iccid #ICCID\_OP\_PROF1,   profileState disabled or enabled,  enabledOnEsimPort 0x00  },  {  iccid #ICCID\_OP\_PROF2,   profileState disabled or enabled,  enabledOnEsimPort 0x01  },  {  iccid #ICCID\_OP\_PROF3,   profileState disabled  } |
| R\_AUTH\_SERVER\_INV\_CERT | resp AuthenticateServerResponse ::= authenticateResponseError : {  transactionId <S\_TRANSACTION\_ID>,  authenticateErrorCode invalidCertificate  } |
| R\_AUTH\_SERVER\_INV\_CHALLENGE | resp AuthenticateServerResponse ::= authenticateResponseError : {  transactionId <S\_TRANSACTION\_ID>,  authenticateErrorCode euiccChallengeMismatch  } |
| R\_AUTH\_SERVER\_INV\_CI | resp AuthenticateServerResponse ::= authenticateResponseError : {  transactionId <S\_TRANSACTION\_ID>,  authenticateErrorCode ciPKUnknown  } |
| R\_AUTH\_SERVER\_INV\_CURV | resp AuthenticateServerResponse ::= authenticateResponseError : {  transactionId <S\_TRANSACTION\_ID>,  authenticateErrorCode unsupportedCurve  } |
| R\_AUTH\_SERVER\_INV\_OID | resp AuthenticateServerResponse ::= authenticateResponseError : {  transactionId <S\_TRANSACTION\_ID>,  authenticateErrorCode invalidOid  } |
| R\_AUTH\_SERVER\_INV\_SIGN | resp AuthenticateServerResponse ::= authenticateResponseError : {  transactionId <S\_TRANSACTION\_ID>,  authenticateErrorCode invalidSignature  } |
| R\_AUTH\_SERVER\_NO\_SESSION | resp AuthenticateServerResponse ::= authenticateResponseError : {  transactionId <S\_TRANSACTION\_ID>,  authenticateErrorCode noSessionContext  } |
| R\_CANCEL\_SESSION\_INV\_TRANS\_ID | resp CancelSessionResponse ::= cancelSessionResponseError : invalidTransactionId |
| R\_CANCEL\_SESSION\_LOAD\_BPP | resp CancelSessionResponse ::= cancelSessionResponseOk : {  euiccCancelSessionSigned {  transactionId <S\_TRANSACTION\_ID>,  smdpOid #S\_SM\_DP+\_OID,  reason loadBppExecutionError  },  euiccCancelSessionSignature <EUICC\_CS\_SIGNATURE>  } |
| R\_CANCEL\_SESSION\_METADATA | resp CancelSessionResponse ::= cancelSessionResponseOk : {  euiccCancelSessionSigned {  transactionId <S\_TRANSACTION\_ID>,  smdpOid #S\_SM\_DP+\_OID,  reason metadataMismatch  },  euiccCancelSessionSignature <EUICC\_CS\_SIGNATURE>  } |
| R\_CANCEL\_SESSION\_POSTPONED | resp CancelSessionResponse ::= cancelSessionResponseOk : {  euiccCancelSessionSigned {  transactionId <S\_TRANSACTION\_ID>,  smdpOid #S\_SM\_DP+\_OID,  reason postponed  },  euiccCancelSessionSignature <EUICC\_CS\_SIGNATURE>  } |
| R\_CANCEL\_SESSION\_PPR | resp CancelSessionResponse ::= cancelSessionResponseOk : {  euiccCancelSessionSigned {  transactionId <S\_TRANSACTION\_ID>,  smdpOid #S\_SM\_DP+\_OID,  reason pprNotAllowed  },  euiccCancelSessionSignature <EUICC\_CS\_SIGNATURE>  } |
| R\_CANCEL\_SESSION\_REJ | resp CancelSessionResponse ::= cancelSessionResponseOk : {  euiccCancelSessionSigned {  transactionId <S\_TRANSACTION\_ID>,  smdpOid #S\_SM\_DP+\_OID,  reason endUserRejection  },  euiccCancelSessionSignature <EUICC\_CS\_SIGNATURE>  } |
| R\_CANCEL\_SESSION\_TIMEOUT | resp CancelSessionResponse ::= cancelSessionResponseOk {  euiccCancelSessionSigned {  transactionId <S\_TRANSACTION\_ID>,  smdpOid #S\_SM\_DP+\_OID,  reason timeout  },  euiccCancelSessionSignature <EUICC\_CS\_SIGNATURE>  } |
| R\_CANCEL\_SESSION\_UNDEF | resp CancelSessionResponse ::= cancelSessionResponseOk : {  euiccCancelSessionSigned {  transactionId <S\_TRANSACTION\_ID>,  smdpOid #S\_SM\_DP+\_OID,  reason undefinedReason  },  euiccCancelSessionSignature <EUICC\_CS\_SIGNATURE>  } |
| R\_CHALLENGE | response GetEuiccChallengeResponse ::=  {  euiccChallenge <EUICC\_CHALLENGE>  } |
| R\_CONF\_OP\_PROF1 | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  isdpAid <ISD\_P\_AID>,  dpProprietaryData {  dpOid #S\_SM\_DP+\_OID,  additionalSmdpData  #ADDITIONAL\_SMDP\_DATA\_MAX\_LENGTH  }  }  }  -- NOTE: Instead of  DpProprietaryData ::= SEQUENCE {  dpOid OBJECT IDENTIFIER  -- additional data objects defined by the  -- SM-DP+ MAY follow  }  -- the following structure is used to test the  -- DpProprietaryData size:  DpProprietaryData ::= SEQUENCE {  dpOid OBJECT IDENTIFIER,  additionalSmdpData OCTET STRING OPTIONAL  } |
| R\_DEFAULT\_RAT | response GetRatResponse ::= {  rat {  #PPRS\_ALLOWED  }  } |
| R\_DELETE\_PROFILE\_DISALLOWEDBYPOLICY | respDelProf DeleteProfileResponse ::= {  deleteResult disallowedByPolicy  } |
| R\_DELETE\_PROFILE\_ICCID\_ISDP\_NOTFOUND | resp DeleteProfileResponse ::= {  deleteResult iccidOrAidNotFound  } |
| R\_DELETE\_PROFILE\_NOTDISABLESTATE | respDelProf DeleteProfileResponse ::= {  deleteResult profileNotInDisabledState  } |
| R\_DELETE\_PROFILE\_OK | respDelProf DeleteProfileResponse ::= {  deleteResult ok  } |
| R\_DISABLE\_PROFILE\_DISALLOWEDbyPOLICY | resp DisableProfileResponse ::= {  disableResult disallowedByPolicy  } |
| R\_DISABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND | resp DisableProfileResponse ::= {  disableResult iccidOrAidNotFound  } |
| R\_DISABLE\_PROFILE\_NOT\_ENABLE\_STATE | resp DisableProfileResponse ::= {  disableResult profileNotInEnabledState  } |
| R\_DISABLE\_PROFILE\_OK | resp DisableProfileResponse ::= {  disableResult ok  } |
| R\_ENABLE\_PROFILE\_DISALLOWEDbyPOLICY | respEnaPro EnableProfileResponse ::= {  enableResult disallowedByPolicy  } |
| R\_ENABLE\_PROFILE\_ICCID\_ISDP\_NOTFOUND | respEnaPro EnableProfileResponse ::= {  enableResult iccidOrAidNotFound  } |
| R\_ENABLE\_PROFILE\_MEP\_A2\_OK | resp EnableProfileResponse ::= {  enableResult ok,  targetEsimPort <MEP-A2\_TARGET\_ESIM\_PORT> } |
| R\_ENABLE\_PROFILE\_NOT\_DISABLE\_STATE | respEnaPro EnableProfileResponse ::= {  enableResult profileNotInDisabledState  } |
| R\_ENABLE\_PROFILE\_OK | resp EnableProfileResponse ::= {  enableResult ok  } |
| R\_ES10a\_GECA\_DS | response EuiccConfiguredDataResponse ::= {  …  -- defaultDpAddress SHALL not be present  rootDsAddress #TEST\_ROOT\_DS\_ADDRESS  } |
| R\_ES10a\_GECA\_DS\_DP\_1 | response EuiccConfiguredDataResponse ::= {  …  defaultDpAddress #TEST\_DP\_ADDRESS1,  rootDsAddress #TEST\_ROOT\_DS\_ADDRESS,  …  } |
| R\_ES10a\_GECA\_DS\_DP\_2 | response EuiccConfiguredDataResponse ::= {  …  defaultDpAddress #TEST\_DP\_ADDRESS2,  rootDsAddress #TEST\_ROOT\_DS\_ADDRESS,  …  } |
| R\_ES10a\_SD\_DP\_A\_OK | response SetDefaultDpAddressResponse::= {  setDefaultDpAddressResult ok  } |
| R\_EUICC\_INFO1 | response EUICCInfo1::=  {  lowestSvn #RSP\_VERSION\_LOWEST\_H,  euiccCiPKIdListForVerification  <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION>,  euiccCiPKIdListForSigning  <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING>,  euiccCiPKIdListForSigningV3  <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3>, --OPTIONAL  euiccRspCapability <EUICC\_RSP\_CAPABILITY>,  highestSvn #IUT\_RSP\_VERSION\_HIGHEST} |
| R\_EUICC\_INFO2 | response EUICCInfo2::=  {  baseProfilePackageVersion #IUT\_SIMA\_VERSION,  lowestSvn #RSP\_VERSION\_LOWEST\_H,  euiccFirmwareVersion #IUT\_EUICC\_FIRMWARE\_VER,  extCardResource <EXT\_CARD\_RESOURCE>,  uiccCapability #IUT\_UICC\_CAPABILITY,  ts102241Version #IUT\_TS102241\_VERSION, --OPTIONAL  globalplatformVersion  #IUT\_GLOBALPLATFORM\_VERSION,  euiccRspCapability <EUICC\_RSP\_CAPABILITY>,  euiccCiPKIdListForVerification  <EUICC\_CI\_PK\_ID\_LIST\_FOR\_VERIFICATION>,  euiccCiPKIdListForSigning  <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING>,  euiccCategory #IUT\_EUICC\_CATEGORY,  forbiddenProfilePolicyRules <PPR\_IDS>, -- OPTIONAL  ppVersion #IUT\_PP\_VERSION,  sasAcreditationNumber #IUT\_SAS\_ACCREDITATION\_NUMBER,  certificationDataObject {  platformLabel #IUT\_PLATFORM\_LABEL,  discoveryBaseURL #IUT\_DLOA\_URL  },  treProperties #IUT\_TRE\_PROPERTIES,  treProductReference #IUT\_TRE\_REFERENCE, -- OPTIONAL  additionalProfilePackageVersions #IUT\_EUICC\_ADD\_PP\_VERSIONS, -- OPTIONAL  lpaMode {lpad},  euiccCiPKIdListForSigningV3  <EUICC\_CI\_PK\_ID\_LIST\_FOR\_SIGNING\_V3>,  additionalEuiccInfo #IUT\_ADDITIONAL\_EUICC\_INFO, --OPTIONAL  highestSvn #IUT\_RSP\_VERSION\_HIGHEST} |
| R\_EUICC\_MEMORY\_RESET\_OK | resp EuiccMemoryResetResponse ::= {  resetResult ok  } |
| R\_GET\_METADATA\_OP\_PROF1 | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  notificationConfigurationInfo {  { profileManagementOperation {  notificationInstall,  notificationLocalEnable,  notificationLocalDisable,  notificationLocalDelete  },  notificationAddress #TEST\_DP\_ADDRESS1  }  },  profileOwner {  mccMnc #MCC\_MNC1  },  profilePolicyRules {ppr1}  }  } |
| R\_GET\_METADATA\_OP\_PROF1\_NO\_SERVICE\_SPECIFIC | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  notificationConfigurationInfo {  { profileManagementOperation {  notificationInstall,  notificationLocalEnable,  notificationLocalDisable,  notificationLocalDelete  },  notificationAddress #TEST\_DP\_ADDRESS1  }  },  profileOwner {  mccMnc #MCC\_MNC1  }  }  } |
| R\_GET\_METADATA\_OP\_PROF1\_SERVICE\_SPECIFIC | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileClass operational,  notificationConfigurationInfo {  { profileManagementOperation {  notificationInstall,  notificationLocalEnable,  notificationLocalDisable,  notificationLocalDelete  },  notificationAddress #TEST\_DP\_ADDRESS1  }  },  profileOwner {  mccMnc #MCC\_MNC1  },  serviceSpecificDataStoredInEuicc #VENDOR\_SPECIFIC\_EXTENSION1  }  } |
| R\_GET\_PROF\_NOTIF\_CONF | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  iccid #ICCID\_OP\_PROF1,  notificationConfigurationInfo {  { profileManagementOperation {  notificationInstall  },  notificationAddress #TEST\_DP\_ADDRESS3  },  { profileManagementOperation {  notificationInstall  },  notificationAddress #TEST\_DP\_ADDRESS2  },  { profileManagementOperation {  notificationLocalEnable  },  notificationAddress #TEST\_DP\_ADDRESS2  },  { profileManagementOperation {  notificationLocalEnable  },  notificationAddress #TEST\_DP\_ADDRESS3  },  { profileManagementOperation {  notificationLocalDisable  },  notificationAddress #TEST\_DP\_ADDRESS3  },  { profileManagementOperation {  notificationLocalDisable  },  notificationAddress #TEST\_DP\_ADDRESS4  },  { profileManagementOperation {  notificationLocalDelete  },  notificationAddress #TEST\_DP\_ADDRESS1  },  { profileManagementOperation {  notificationLocalDelete  },  notificationAddress #TEST\_DP\_ADDRESS3  }  }  }  } |
| R\_GET\_UPDATE\_N1 | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profilePolicyRules { ppr2 }  }  } |
| R\_GET\_UPDATE\_N2 | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType jpg,  icon #ICON\_JPG,  profilePolicyRules { ppr1 }  }  } |
| R\_GET\_UPDATE\_N3 | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  serviceProviderName #SP\_NAME2,  profileName #NAME\_OP\_PROF2,  iconType png,  icon #ICON\_OP\_PROF1  -- profilePolicyRules SHALL not be present  }  } |
| R\_GET\_UPDATE\_N6 | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  serviceProviderName #SP\_NAME2,  profileName #NAME\_OP\_PROF2,  iconType png,  icon #ICON\_OP\_PROF1  -- profilePolicyRules SHALL not be present  }  } |
| R\_GET\_UPDATE\_N7 | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  serviceProviderName #SP\_NAME2,  profileName #NAME\_OP\_PROF2  -- iconType SHALL not be present  -- icon SHALL not be present  -- profilePolicyRules SHALL not be present  }  } |
| R\_GET\_UPDATE\_N8 | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  notificationConfigurationInfo {  { profileManagementOperation {  notificationLocalDelete  },  notificationAddress #TEST\_DP\_ADDRESS2  }  },  profilePolicyRules { ppr1, ppr2 }  }  } |
| R\_GET\_UPDATE\_N9 | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profilePolicyRules { ppr1, ppr2 },  }  } |
| R\_GET\_UPDATE\_N10 | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profilePolicyRules { ppr1, ppr2 },  serviceSpecificDataStoredInEuicc #VENDOR\_SPECIFIC\_EXTENSION2,  }  } |
| R\_GET\_UPDATE\_N11 | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profilePolicyRules { ppr1, ppr2 }  }  } |
| R\_GET\_UPDATE\_N12 | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profilePolicyRules { ppr1, ppr2 },  rpmConfiguration {  managingDpList {},  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  }  } |
| R\_GET\_UPDATE\_N13 | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profilePolicyRules { ppr1, ppr2 },  }  } |
| R\_GET\_UPDATE\_N14 | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profilePolicyRules { ppr1, ppr2 },  hriServerAddress #TEST\_HRI\_ADDRESS3  }  } |
| R\_GET\_UPDATE\_N15 | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profilePolicyRules { ppr1, ppr2 },  lprConfiguration {  pcmpAddress #TEST\_PCMP\_ADDRESS3  }  }  } |
| R\_GET\_UPDATE\_N16 | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profilePolicyRules { ppr1, ppr2  }  } |
| R\_GET\_UPDATE\_N17 | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profilePolicyRules { ppr1, ppr2 },  DeviceChangeConfiguration {  requestToDp {  smdpAddressToBeUsedForDc #TEST\_DP\_ADDRESS2  }  }  }  } |
| R\_GET\_UPDATE\_N18 | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profilePolicyRules { ppr1, ppr2 }  }  } |
| R\_GET\_UPDATE\_N19 | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profilePolicyRules { ppr1, ppr2 },  DeviceChangeConfiguration {  usingStoredAc {  activationCodeForDc ACTIVATION\_CODE\_2  }  }  }  } |
| R\_GET\_UPDATE\_N20 | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profilePolicyRules { ppr1, ppr2 }  }  } |
| R\_ISDR\_SELECTION | resp ISDRProprietaryApplicationTemplate::= {  lowestSvn #RSP\_VERSION\_LOWEST\_H,  euiccConfiguration {}  } |
| R\_ISDR\_SELECTION\_EN\_PROF | resp ISDRProprietaryApplicationTemplate::= {  lowestSvn #RSP\_VERSION\_LOWEST\_H,  euiccConfiguration {  ...,  enabledProfile  }  } |
| R\_ISDR\_SELECTION\_LPAE | resp ISDRProprietaryApplicationTemplate::= {  lowestSvn #RSP\_VERSION\_LOWEST\_H,  euiccConfiguration {  lpaeUsingCatSupported  OR  lpaeUsingScwsSupported  }  } |
| R\_LIST\_NOTIF\_DE1 | response ListNotificationResponse ::= notificationMetadataList : {  #NOTIF\_METADATA\_DELETE1  } |
| R\_LIST\_NOTIF\_DE1\_DE1 | response ListNotificationResponse ::= notificationMetadataList : {  #NOTIF\_METADATA\_DELETE1,  #NOTIF\_METADATA2\_DELETE1  } |
| R\_LIST\_NOTIF\_DE1\_DE2 | response ListNotificationResponse ::= notificationMetadataList : {  #NOTIF\_METADATA\_DELETE1,  #NOTIF\_METADATA\_DELETE2  } |
| R\_LIST\_NOTIF\_DI1 | response ListNotificationResponse ::= notificationMetadataList : {  #NOTIF\_METADATA\_DISABLE1  } |
| R\_LIST\_NOTIF\_DI1\_DE1 | response ListNotificationResponse ::= notificationMetadataList : {  #NOTIF\_METADATA\_DISABLE1,  #NOTIF\_METADATA\_DELETE1  } |
| R\_LIST\_NOTIF\_DI1\_DI1 | response ListNotificationResponse ::= notificationMetadataList : {  #NOTIF\_METADATA\_DISABLE1,  #NOTIF\_METADATA2\_DISABLE1  } |
| R\_LIST\_NOTIF\_DI1\_EN2 | response ListNotificationResponse ::= notificationMetadataList : {  #NOTIF\_METADATA\_DISABLE1,  #NOTIF\_METADATA\_ENABLE2  } |
| R\_LIST\_NOTIF\_EN1 | response ListNotificationResponse ::= notificationMetadataList: {  #NOTIF\_METADATA\_ENABLE1  } |
| R\_LIST\_NOTIF\_EN1\_EN1 | response ListNotificationResponse ::= notificationMetadataList : {  #NOTIF\_METADATA\_ENABLE1,  #NOTIF\_METADATA2\_ENABLE1  } |
| R\_LIST\_NOTIF\_EN1\_IN2\_PIR | response ListNotificationResponse ::= notificationMetadataList: {  #NOTIF\_METADATA\_ENABLE1,  #NOTIF\_METADATA\_INSTALL2\_PIR  } |
| R\_LIST\_NOTIF\_EN1\_RPM | response ListNotificationResponse ::=  notificationMetadataList: {  #NOTIF\_METADATA\_EN1\_RPM  } |
| R\_LIST\_NOTIF\_EN1\_RPM\_DP2 | response ListNotificationResponse ::=  notificationMetadataList: {  #NOTIF\_METADATA\_DP2\_EN1\_RPM  } |
| R\_LIST\_NOTIF\_IN1 | response ListNotificationResponse ::= notificationMetadataList: {  #NOTIF\_METADATA\_INSTALL1  } |
| R\_LIST\_NOTIF\_IN1\_DP1\_PIR\_IN1\_DP2\_OSN | response ListNotificationResponse ::= notificationMetadataList: {  #NOTIF\_METADATA\_INSTALL1\_DP1\_PIR,  #NOTIF\_METADATA\_INSTALL1\_DP2\_OSN  } |
| R\_LIST\_NOTIF\_IN1\_EN1 | response ListNotificationResponse ::= notificationMetadataList: {  #NOTIF\_METADATA\_INSTALL1,  #NOTIF\_METADATA\_ENABLE1  } |
| R\_LIST\_NOTIF\_IN1\_IN1\_PIR | response ListNotificationResponse ::= notificationMetadataList: {  #NOTIF\_METADATA\_INSTALL1,  #NOTIF\_METADATA\_INSTALL1\_PIR  } |
| R\_LIST\_NOTIF\_IN1\_PIR | response ListNotificationResponse ::= notificationMetadataList: {  #NOTIF\_METADATA\_INSTALL1\_PIR  } |
| R\_LIST\_NOTIF\_IN1\_PIR\_EN1 | response ListNotificationResponse ::= notificationMetadataList: {  #NOTIF\_METADATA\_INSTALL1\_PIR,  #NOTIF\_METADATA\_ENABLE1  } |
| R\_LIST\_NOTIF\_IN1\_PIR\_IN2\_PIR | response ListNotificationResponse ::= notificationMetadataList: {  #NOTIF\_METADATA\_INSTALL1\_PIR,  #NOTIF\_METADATA\_INSTALL2\_PIR  } |
| R\_LIST\_NOTIF\_IN2\_PIR | response ListNotificationResponse ::= notificationMetadataList: {  #NOTIF\_METADATA\_INSTALL2\_PIR  } |
| R\_LIST\_NOTIF\_NONE | response ListNotificationResponse ::=  notificationMetadataList: {} |
| R\_LIST\_NOTIF\_RPR | response ListNotificationResponse ::=  notificationMetadataList: {  #NOTIF\_METADATA\_PROF1\_DP1\_RPR  } |
| R\_LIST\_NOTIF\_RPR\_EN1\_RPM\_DP2 | response ListNotificationResponse ::=  notificationMetadataList: {  #NOTIF\_METADATA\_PROF1\_DP1\_RPR,  #NOTIF\_METADATA\_DP2\_EN1\_RPM  } |
| R\_LIST\_NOTIF\_UNDEFINED\_ERROR | response ListNotificationResponse ::= listNotificationsResultError : undefinedError |
| R\_METADATA\_UNCHANGED | resp ProfileInfoListResponse ::=  profileInfoListOk :{  {  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profilePolicyRules {ppr1,ppr2}  }  } |
| R\_PIR\_DATA\_MISMATCH | resp ProfileInstallationResult ::= {  profileInstallationResultData {  transactionId <S\_TRANSACTION\_ID>,  notificationMetadata {  seqNumber <SEQ\_NUMBER>,  profileManagementOperation {  notificationInstall  },  notificationAddress #TEST\_DP\_ADDRESS1,  ...  },  smdpOid #S\_SM\_DP+\_OID,  finalResult errorResult : {  bppCommandId loadProfileElements,  errorReason installFailedDueToDataMismatch,  ...  }  },  euiccSignPIR <EUICC\_SIGN\_PIR>  } |
| R\_PIR\_EOID\_MISMATCH | resp ProfileInstallationResult ::= {  profileInstallationResultData {  transactionId <S\_TRANSACTION\_ID>,  notificationMetadata {  seqNumber <SEQ\_NUMBER>,  profileManagementOperation {  notificationInstall  },  notificationAddress #TEST\_DP\_ADDRESS1  },  smdpOid #S\_SM\_DP+\_OID,  finalResult errorResult : {  bppCommandId storeMetadata,  errorReason  enterpriseOidMismatch  }  },  euiccSignPIR <EUICC\_SIGN\_PIR>  } |
| R\_PIR\_EP\_NOTALLOWED | resp ProfileInstallationResult ::= {  profileInstallationResultData {  transactionId <S\_TRANSACTION\_ID>,  notificationMetadata {  seqNumber <SEQ\_NUMBER>,  profileManagementOperation {  notificationInstall  },  notificationAddress #TEST\_DP\_ADDRESS1  },  smdpOid #S\_SM\_DP+\_OID,  finalResult errorResult : {  bppCommandId storeMetadata,  errorReason  enterpriseProfileNotAllowed  }  },  euiccSignPIR <EUICC\_SIGN\_PIR>  } |
| R\_PIR\_EP\_ONLY | resp ProfileInstallationResult ::= {  profileInstallationResultData {  transactionId <S\_TRANSACTION\_ID>,  notificationMetadata {  seqNumber <SEQ\_NUMBER>,  profileManagementOperation {  notificationInstall  },  notificationAddress #TEST\_DP\_ADDRESS1  },  smdpOid #S\_SM\_DP+\_OID,  finalResult errorResult : {  bppCommandId storeMetadata,  errorReason  enterpriseProfilesOnly  }  },  euiccSignPIR <EUICC\_SIGN\_PIR>  } |
| R\_PIR\_ER\_NOTALLOWED | resp ProfileInstallationResult ::= {  profileInstallationResultData {  transactionId <S\_TRANSACTION\_ID>,  notificationMetadata {  seqNumber <SEQ\_NUMBER>,  profileManagementOperation {  notificationInstall  },  notificationAddress #TEST\_DP\_ADDRESS1  },  smdpOid #S\_SM\_DP+\_OID,  finalResult errorResult : {  bppCommandId storeMetadata,  errorReason  enterpriseRulesNotAllowed  }  },  euiccSignPIR <EUICC\_SIGN\_PIR>  } |
| R\_PIR\_ICCID\_ALREADY\_EXIST | resp ProfileInstallationResult ::= {  profileInstallationResultData {  transactionId <S\_TRANSACTION\_ID>,  notificationMetadata {  seqNumber <SEQ\_NUMBER>,  profileManagementOperation {  notificationInstall  },  notificationAddress #TEST\_DP\_ADDRESS1,  iccid #ICCID\_OP\_PROF1  },  smdpOid #S\_SM\_DP+\_OID,  finalResult errorResult : {  bppCommandId storeMetadata,  errorReason  installFailedDueToIccidAlreadyExistsOnEuicc  }  },  euiccSignPIR <EUICC\_SIGN\_PIR>  } |
| R\_PIR\_INVALID\_CRT | resp ProfileInstallationResult ::= {  profileInstallationResultData {  transactionId <S\_TRANSACTION\_ID>,  notificationMetadata {  seqNumber <SEQ\_NUMBER>,  profileManagementOperation {  notificationInstall  },  notificationAddress #TEST\_DP\_ADDRESS1  },  smdpOid #S\_SM\_DP+\_OID,  finalResult errorResult : {  bppCommandId initialiseSecureChannel,  errorReason unsupportedCrtValues  }  },  euiccSignPIR <EUICC\_SIGN\_PIR>  } |
| R\_PIR\_INVALID\_DATA | resp ProfileInstallationResult ::= {  profileInstallationResultData {  transactionId <S\_TRANSACTION\_ID>,  notificationMetadata {  seqNumber <SEQ\_NUMBER>,  profileManagementOperation {  notificationInstall  },  notificationAddress #TEST\_DP\_ADDRESS1  },  smdpOid #S\_SM\_DP+\_OID,  finalResult errorResult : {  bppCommandId configureISDP,  errorReason incorrectInputValues  }  },  euiccSignPIR <EUICC\_SIGN\_PIR>  } |
| R\_PIR\_INVALID\_OP\_ID | resp ProfileInstallationResult ::= {  profileInstallationResultData {  transactionId <S\_TRANSACTION\_ID>,  notificationMetadata {  seqNumber <SEQ\_NUMBER>,  profileManagementOperation {  notificationInstall  },  notificationAddress #TEST\_DP\_ADDRESS1  },  smdpOid #S\_SM\_DP+\_OID,  finalResult errorResult : {  bppCommandId initialiseSecureChannel,  errorReason unsupportedRemoteOperationType  }  },  euiccSignPIR <EUICC\_SIGN\_PIR>  } |
| R\_PIR\_INVALID\_SIGN | resp ProfileInstallationResult ::= {  profileInstallationResultData {  transactionId <S\_TRANSACTION\_ID>,  notificationMetadata {  seqNumber <SEQ\_NUMBER>,  profileManagementOperation {  notificationInstall  },  notificationAddress #TEST\_DP\_ADDRESS1  },  smdpOid #S\_SM\_DP+\_OID,  finalResult errorResult : {  bppCommandId initialiseSecureChannel,  errorReason invalidSignature  }  },  euiccSignPIR <EUICC\_SIGN\_PIR>  } |
| R\_PIR\_INVALID\_TRANS\_ID | resp ProfileInstallationResult ::= {  profileInstallationResultData {  transactionId <INVALID\_TRANSACTION\_ID>,  notificationMetadata {  seqNumber <SEQ\_NUMBER>,  profileManagementOperation {  notificationInstall  },  notificationAddress #TEST\_DP\_ADDRESS1  },  smdpOid #S\_SM\_DP+\_OID,  finalResult errorResult : {  bppCommandId initialiseSecureChannel,  errorReason invalidTransactionId  }  },  euiccSignPIR <EUICC\_SIGN\_PIR>  } |
| R\_PIR\_METADATA\_INVALID | resp ProfileInstallationResult ::= {  profileInstallationResultData {  transactionId <S\_TRANSACTION\_ID>,  notificationMetadata {  seqNumber <SEQ\_NUMBER>,  profileManagementOperation {  notificationInstall  },  notificationAddress #TEST\_DP\_ADDRESS1,  ...  },  smdpOid #S\_SM\_DP+\_OID,  finalResult errorResult : {  bppCommandId storeMetadata,  errorReason  bspStructureError  OR  incorrectInputValues  }  },  euiccSignPIR <EUICC\_SIGN\_PIR>  } |
| R\_PIR\_OK | response ProfileInstallationResult ::= {  profileInstallationResultData {  transactionId <S\_TRANSACTION\_ID>,  notificationMetadata {  seqNumber <SEQ\_NUMBER>,  profileManagementOperation {  notificationInstall  },  notificationAddress #TEST\_DP\_ADDRESS1,  iccid #ICCID\_OP\_PROF1  },  smdpOid #S\_SM\_DP+\_OID,  finalResult successResult : {  aid <ISD\_P\_AID>,  simaResponse #SIMA\_RESULT\_OK  }  },  euiccSignPIR <EUICC\_SIGN\_PIR>  } |
| R\_PIR\_OK\_PROF9 | response ProfileInstallationResult ::= {  profileInstallationResultData {  transactionId <S\_TRANSACTION\_ID>,  notificationMetadata {  seqNumber <SEQ\_NUMBER>,  profileManagementOperation {  notificationInstall  },  notificationAddress #TEST\_DP\_ADDRESS1,  iccid #ICCID\_OP\_PROF9  },  smdpOid #S\_SM\_DP+\_OID,  finalResult successResult : {  aid <ISD\_P\_AID>,  simaResponse #SIMA\_RESULT\_OK  }  },  euiccSignPIR <EUICC\_SIGN\_PIR>  } |
| R\_PIR\_OK\_PROF10 | response ProfileInstallationResult ::= {  profileInstallationResultData {  transactionId <S\_TRANSACTION\_ID>,  notificationMetadata {  seqNumber <SEQ\_NUMBER>,  profileManagementOperation {  notificationInstall  },  notificationAddress #TEST\_DP\_ADDRESS1,  iccid #ICCID\_OP\_PROF10  },  smdpOid #S\_SM\_DP+\_OID,  finalResult successResult : {  aid <ISD\_P\_AID>,  simaResponse #SIMA\_RESULT\_OK  }  },  euiccSignPIR <EUICC\_SIGN\_PIR>  } |
| R\_PIR\_PPK\_INV | resp ProfileInstallationResult ::= {  profileInstallationResultData {  ...  finalResult errorResult : {  bppCommandId replaceSessionKeys,  errorReason  incorrectInputValues  OR  bspStructureError  OR  bspSecurityError  }  },  euiccSignPIR <EUICC\_SIGN\_PIR>  } |
| R\_PIR\_PPR\_NOT\_ALLOWED | resp ProfileInstallationResult ::= {  profileInstallationResultData {  transactionId <S\_TRANSACTION\_ID>,  notificationMetadata {  seqNumber <SEQ\_NUMBER>,  profileManagementOperation {  notificationInstall  },  notificationAddress #TEST\_DP\_ADDRESS1,  ...  },  smdpOid #S\_SM\_DP+\_OID,  finalResult errorResult : {  bppCommandId storeMetadata,  errorReason pprNotAllowed  }  },  euiccSignPIR <EUICC\_SIGN\_PIR>  } |
| R\_PIR\_RER\_NOTALLOWED | resp ProfileInstallationResult ::= {  profileInstallationResultData {  transactionId <S\_TRANSACTION\_ID>,  notificationMetadata {  seqNumber <SEQ\_NUMBER>,  profileManagementOperation {  notificationInstall  },  notificationAddress #TEST\_DP\_ADDRESS1  },  smdpOid #S\_SM\_DP+\_OID,  finalResult errorResult : {  bppCommandId storeMetadata,  errorReason  enterpriseRulesError  }  },  euiccSignPIR <EUICC\_SIGN\_PIR>  } |
| R\_PIR\_SECU\_INVALID | resp ProfileInstallationResult ::= {  profileInstallationResultData {  transactionId <S\_TRANSACTION\_ID>,  ...  smdpOid #S\_SM\_DP+\_OID,  finalResult errorResult : {  bppCommandId loadProfileElements,  errorReason incorrectInputValues  OR  bspStructureError  OR  bspSecurityError  ...  }  },  euiccSignPIR <EUICC\_SIGN\_PIR>  } |
| R\_PIR\_UNKNOWN\_TLV | resp ProfileInstallationResult ::= {  profileInstallationResultData {  transactionId <S\_TRANSACTION\_ID>,  notificationMetadata {  seqNumber <SEQ\_NUMBER>,  profileManagementOperation {  notificationInstall  },  notificationAddress #TEST\_DP\_ADDRESS1  },  smdpOid #S\_SM\_DP+\_OID,  finalResult errorResult : {  bppCommandId storeMetadata,  errorReason  unknownTlvInMetadata  }  },  euiccSignPIR <EUICC\_SIGN\_PIR>  } |
| R\_PREP\_DOWN\_INV\_TRANS\_ID | resp PrepareDownloadResponse ::= downloadResponseError : {  transactionId <INVALID\_TRANSACTION\_ID>,  downloadErrorCode invalidTransactionId  } |
| R\_PREP\_DOWN\_NO\_SESSION | resp PrepareDownloadResponse ::= downloadResponseError : {  transactionId <S\_TRANSACTION\_ID>,  downloadErrorCode noSession  } |
| R\_PREP\_DOWNLOAD\_INV\_CERT | resp PrepareDownloadResponse ::= downloadResponseError : {  transactionId <S\_TRANSACTION\_ID>,  downloadErrorCode invalidCertificate  } |
| R\_PREP\_DOWNLOAD\_INV\_SIGN | resp PrepareDownloadResponse ::= downloadResponseError : {  transactionId <S\_TRANSACTION\_ID>,  downloadErrorCode invalidSignature  } |
| R\_PREP\_DOWNLOAD\_NO\_CC | resp PrepareDownloadResponse ::= downloadResponseOk : {  euiccSigned2 {  transactionId <S\_TRANSACTION\_ID>,  euiccOtpk <OTPK\_EUICC\_ECKA>  },  euiccSignature2 <EUICC\_SIGNATURE2>  } |
| R\_PREP\_DOWNLOAD\_WITH\_CC | resp PrepareDownloadResponse ::= downloadResponseOk : {  euiccSigned2 {  transactionId <S\_TRANSACTION\_ID>,  euiccOtpk <OTPK\_EUICC\_ECKA>,  hashCc <S\_HASHED\_CC>  },  euiccSignature2 <EUICC\_SIGNATURE2>  } |
| R\_RAT\_WITH\_OTHER\_RULES | response GetRatResponse ::= {  rat {  #PPR1\_WITH\_OWNER\_GID,  #PPR1\_WITHOUT\_GID,  #PPR2\_WITHOUT\_CONSENT,  #PPRS\_ALLOWED  }  } |
| R\_REMOVE\_NOTIF\_NOTHING\_TO\_DELETE | response NotificationSentResponse ::= {  deleteNotificationStatus nothingToDelete  } |
| R\_REMOVE\_NOTIF\_OK | response NotificationSentResponse ::= {  deleteNotificationStatus ok  } |
| R\_RETRIEVE\_NOTIF\_DE1\_V3 | resp RetrieveNotificationsListResponse ::=  notificationList : {  otherSignedNotification : {  tbsOtherNotification #NOTIF\_METADATA\_DELETE1,  euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG>,  euiccCertificate #CERT\_EUICC\_SIG,  nextCertInChain #CERT\_EUM\_SIG  }  } |
| R\_RETRIEVE\_NOTIF\_DI1\_V3 | resp RetrieveNotificationsListResponse ::=  notificationList : {  otherSignedNotification : {  tbsOtherNotification #NOTIF\_METADATA\_DISABLE1,  euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG>,  euiccCertificate #CERT\_EUICC\_SIG,  nextCertInChain #CERT\_EUM\_SIG  }  } |
| R\_RETRIEVE\_NOTIF\_DI1\_DE1\_V3 | resp RetrieveNotificationsListResponse ::=  notificationList : {  otherSignedNotification : {  tbsOtherNotification #NOTIF\_METADATA\_DISABLE1,  euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG>,  euiccCertificate #CERT\_EUICC\_SIG,  nextCertInChain #CERT\_EUM\_SIG  },  otherSignedNotification : {  tbsOtherNotification #NOTIF\_METADATA\_DELETE1,  euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG>,  euiccCertificate #CERT\_EUICC\_SIG,  nextCertInChain #CERT\_EUM\_SIG  }  } |
| R\_RETRIEVE\_NOTIF\_EN1\_V3 | resp RetrieveNotificationsListResponse ::=  notificationList : {  otherSignedNotification : {  tbsOtherNotification #NOTIF\_METADATA\_ENABLE1,  euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG>,  euiccCertificate #CERT\_EUICC\_SIG,  nextCertInChain #CERT\_EUM\_SIG  }  } |
| R\_RETRIEVE\_NOTIF\_EN1\_IN2\_PIR\_V3 | resp RetrieveNotificationsListResponse ::=  notificationList : {  profileInstallationResult : {  profileInstallationResultData {  transactionId <S\_TRANSACTION\_ID>,  notificationMetadata #NOTIF\_METADATA\_INSTALL2\_PIR,  smdpOid #S\_SM\_DP+\_OID2,  finalResult successResult : {  aid <ISD\_P\_AID>,  simaResponse #SIMA\_RESULT\_OK  }  },  euiccSignPIR <EUICC\_SIGN\_PIR>  },  otherSignedNotification : {  tbsOtherNotification#NOTIF\_METADATA\_ENABLE1,  euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG>,  euiccCertificate #CERT\_EUICC\_SIG,  nextCertInChain #CERT\_EUM\_SIG  }  } |
| R\_RETRIEVE\_NOTIF\_IN1\_V3 | resp RetrieveNotificationsListResponse ::=  notificationList : {  otherSignedNotification : {  tbsOtherNotification #NOTIF\_METADATA\_INSTALL1,  euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG>,  euiccCertificate #CERT\_EUICC\_SIG,  nextCertInChain #CERT\_EUM\_SIG  }  } |
| R\_RETRIEVE\_NOTIF\_IN1\_DP1\_PIR | resp RetrieveNotificationsListResponse ::=  notificationList : {  profileInstallationResult : {  profileInstallationResultData {  transactionId <S\_TRANSACTION\_ID>,  notificationMetadata #NOTIF\_METADATA\_INSTALL1\_DP1\_PIR,  smdpOid #S\_SM\_DP+\_OID1,  finalResult successResult : {  aid <ISD\_P\_AID>,  simaResponse #SIMA\_RESULT\_OK  }  },  euiccSignPIR <EUICC\_SIGN\_PIR>  }  } |
| R\_RETRIEVE\_NOTIF\_IN1\_DP1\_PIR\_IN1\_DP2\_OSN\_V3 | resp RetrieveNotificationsListResponse ::=  notificationList : {  profileInstallationResult : {  profileInstallationResultData {  transactionId <S\_TRANSACTION\_ID>,  notificationMetadata #NOTIF\_METADATA\_INSTALL1\_DP1\_PIR,  smdpOid #S\_SM\_DP+\_OID1,  finalResult successResult : {  aid <ISD\_P\_AID>,  simaResponse #SIMA\_RESULT\_OK  }  },  euiccSignPIR <EUICC\_SIGN\_PIR>  },  otherSignedNotification : {  tbsOtherNotification #NOTIF\_METADATA\_INSTALL1\_DP2\_OSN,  euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG>,  euiccCertificate #CERT\_EUICC\_SIG,  nextCertInChain #CERT\_EUM\_SIG  }  } |
| R\_RETRIEVE\_NOTIF\_IN1\_DP2\_OSN\_V3 | resp RetrieveNotificationsListResponse ::=  notificationList : {  otherSignedNotification : {  tbsOtherNotification #NOTIF\_METADATA\_INSTALL1\_DP2\_OSN,  euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG>,  euiccCertificate #CERT\_EUICC\_SIG,  nextCertInChain #CERT\_EUM\_SIG  }  } |
| R\_RETRIEVE\_NOTIF\_IN1\_EN1\_V3 | resp RetrieveNotificationsListResponse ::=  notificationList : {  otherSignedNotification : {  tbsOtherNotification #NOTIF\_METADATA\_INSTALL1,  euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG>,  euiccCertificate #CERT\_EUICC\_SIG,  nextCertInChain #CERT\_EUM\_SIG  },  otherSignedNotification : {  tbsOtherNotification #NOTIF\_METADATA\_ENABLE1,  euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG>,  euiccCertificate #CERT\_EUICC\_SIG,  nextCertInChain #CERT\_EUM\_SIG  }  } |
| R\_RETRIEVE\_NOTIF\_IN1\_IN1\_PIR | resp RetrieveNotificationsListResponse ::=  notificationList : {  profileInstallationResult : {  profileInstallationResultData {  transactionId <S\_TRANSACTION\_ID>,  notificationMetadata #NOTIF\_METADATA\_INSTALL1\_PIR,  smdpOid #S\_SM\_DP+\_OID,  finalResult successResult : {  aid <ISD\_P\_AID>,  simaResponse #SIMA\_RESULT\_OK  }  },  euiccSignPIR <EUICC\_SIGN\_PIR>  },  otherSignedNotification : {  tbsOtherNotification #NOTIF\_METADATA\_INSTALL1,  euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG>,  euiccCertificate #CERT\_EUICC\_SIG,  nextCertInChain #CERT\_EUM\_SIG  }  } |
| R\_RETRIEVE\_NOTIF\_IN1\_PIR | resp RetrieveNotificationsListResponse ::=  notificationList : {  profileInstallationResult : {  profileInstallationResultData {  transactionId <S\_TRANSACTION\_ID>,  notificationMetadata #NOTIF\_METADATA\_INSTALL1\_PIR,  smdpOid #S\_SM\_DP+\_OID,  finalResult successResult : {  aid <ISD\_P\_AID>,  simaResponse #SIMA\_RESULT\_OK  }  },  euiccSignPIR <EUICC\_SIGN\_PIR>  }  } |
| R\_RETRIEVE\_NOTIF\_IN1\_PIR\_EN1\_V3 | resp RetrieveNotificationsListResponse ::=  notificationList : {  profileInstallationResult : {  profileInstallationResultData {  transactionId <S\_TRANSACTION\_ID>,  notificationMetadata #NOTIF\_METADATA\_INSTALL1\_PIR,  smdpOid #S\_SM\_DP+\_OID,  finalResult successResult : {  aid <ISD\_P\_AID>,  simaResponse #SIMA\_RESULT\_OK  }  },  euiccSignPIR <EUICC\_SIGN\_PIR>  },  otherSignedNotification : {  tbsOtherNotification #NOTIF\_METADATA\_ENABLE1,  euiccNotificationSignature <TBS\_EUICC\_NOTIF\_SIG>,  euiccCertificate #CERT\_EUICC\_SIG,  nextCertInChain #CERT\_EUM\_SIG  }  } |
| R\_RETRIEVE\_NOTIF\_IN1\_PIR\_IN2\_PIR | resp RetrieveNotificationsListResponse ::=  notificationList : {  profileInstallationResult : {  profileInstallationResultData {  transactionId <S\_TRANSACTION\_ID>,  notificationMetadata #NOTIF\_METADATA\_INSTALL1\_PIR,  smdpOid #S\_SM\_DP+\_OID,  finalResult successResult : {  aid <ISD\_P\_AID>,  simaResponse #SIMA\_RESULT\_OK  }  },  euiccSignPIR <EUICC\_SIGN\_PIR>  },  profileInstallationResult : {  profileInstallationResultData {  transactionId <S\_TRANSACTION\_ID>,  notificationMetadata #NOTIF\_METADATA\_INSTALL2\_PIR,  smdpOid #S\_SM\_DP+\_OID2,  finalResult successResult : {  aid <ISD\_P\_AID>,  simaResponse #SIMA\_RESULT\_OK  }  },  euiccSignPIR <EUICC\_SIGN\_PIR>  }  } |
| R\_RETRIEVE\_NOTIF\_IN2\_PIR | resp RetrieveNotificationsListResponse ::=  notificationList : {  profileInstallationResult : {  profileInstallationResultData {  transactionId <S\_TRANSACTION\_ID>,  notificationMetadata #NOTIF\_METADATA\_INSTALL2\_PIR,  smdpOid #S\_SM\_DP+\_OID2,  finalResult successResult : {  aid <ISD\_P\_AID>,  simaResponse #SIMA\_RESULT\_OK  }  },  euiccSignPIR <EUICC\_SIGN\_PIR>  }  } |
| R\_RETRIEVE\_NOTIF\_NONE | resp RetrieveNotificationsListResponse ::=  notificationList : {} |
| SMDP\_PROP\_DATA1  (DpProprietaryData) | {  dpOid #S\_SM\_DP+\_OID  } |

D.4 APDU

D.4.1 APDU Commands

|  |  |
| --- | --- |
| Name | Content |
| DELETE\_SSD | - CLA = 80, INS = E4, P1 = 00, P2 = 80, LC = <L>  - Data = 4F <L> #SSD\_AID  - LE = 00 |
| GET\_MNO\_SD | - CLA = 80, INS = F2, P1 = 80, P2 = 02, LC = <L>  - Data = 4F 00  - LE = 00 |
| GET\_RESPONSE | - CLA = 0x (x = <CHANNEL\_NUMBER>), INS = C0,  P1 = 00, P2 = 00, LE = <L> |
| INSTALL\_PERSO\_RES\_ISDP | - CLA = 80, INS = E6, P1 = 20, P2 = 00, LC = 16 - Data = 00 00 10 A0 00 00 05 59 10 10 FF FF FF FF 89 00 00 0F 00 00 00 00  - LE = 00 |
| MANAGE\_CHANNEL\_OPEN | - CLA = 00, INS = 70, P1 = 00, P2 = 00, LE = 01 |
| MANAGE\_LSI(Select LSI) | - CLA = 80, INS = 7C, P1 = 00, P2 = <LSI\_NUMBER>, LE = 00 |
| READ\_BINARY | - CLA = 00, INS = B0, P1 = 00, P2 = 00, LE = <L> |
| SELECT\_ICCID | - CLA = 00, INS = A4, P1 = 00, P2 = 0C, LC = 02 - Data = 2F E2 |
| SELECT\_MF | - CLA = 00, INS = A4, P1 = 00, P2 = 04, LC = <L>  - Data = 3F 00  - LE = 00 |
| SELECT\_USIM | - CLA = 00, INS = A4, P1 = 04, P2 = 04, LC = <L> - Data = #USIM\_AID  - LE = 00 |
| TERMINAL\_CAPABILITY\_Enterprise | - CLA = 80, INS = AA, P1 = 00, P2 = 00, LC = <L>  - Data = A9 05 81 00 83 01 27 |
| TERMINAL\_CAPABILITY\_LPA\_Alerting | - CLA = 80, INS = AA, P1 = 00, P2 = 00, LC = <L>  - Data = A9 05 81 00 83 01 17 |
| TERMINAL\_CAPABILITY\_LPAd | - CLA = 80, INS = AA, P1 = 00, P2 = 00, LC = <L>  - Data = A9 05 81 00 83 01 07 |
| TERMINAL\_PROFILE | - CLA = 80, INS = 10, P1 = 00, P2 = 00, LC = <L>  - Data = FF FF FF FF 7F 9D 00 DF BF 00 00 1F E2 00 00 00 C7 EB 00 00 00 01 68 00 50 00 00 00 00 00 02 00 |
| TERMINAL\_PROFILE\_eUICCProfileStateChanged | - CLA = 80, INS = 10, P1 = 00, P2 = 00, LC = <L>  - Data = FF FF FF FF FF FF 1F FF FF 03 02 FF FF 9F FF EF DF FF 0F FF 0F FF FF 0F FF 03 00 3F 7F FF 03 FF FF 20 |
| TERMINAL\_PROFILE\_LSI\_COMMAND | - CLA = 80, INS = 10, P1 = 00, P2 = 00, LC = <L>  - Data = FF FF FF FF FF FF 1F FF FF 03 02 FF FF 9F FF EF DF FF 0F FF 0F FF FF 0F FF 03 00 3F 7F FF 03 FF FF 00 30 00 |
| TERMINAL\_PROFILE\_LSI\_COMMAND\_eUICCProfileStateChanged | - CLA = 80, INS = 10, P1 = 00, P2 = 00, LC = <L>  - Data = FF FF FF FF FF FF 1F FF FF 03 02 FF FF 9F FF EF DF FF 0F FF 0F FF FF 0F FF 03 00 3F 7F FF 03 FF FF 20 30 00 |

D.4.2 R-APDU Chaining

During the execution of all sequences related to the eUICC testing (i.e. section 4.2), for commands where the response exceeds 256 bytes, the chaining mechanism defined in ISO/IEC 7816-4 [7], using the 61XX status word and multiple GET RESPONSE commands, SHALL be used.

As an example, the following generic sequence, which describes this mechanism, SHALL apply.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Direction | Sequence / Description | Result |
| 1 | OCE → eUICC | Send APDU command on logical channel x | <R\_APDU\_PART1>  SW=0x61XX |
| 2 | OCE → eUICC | Send [GET\_RESPONSE] on logical channel x with LE='XX' | <R\_APDU\_PART2>  SW=0x61XX |
| 3 | OCE → eUICC | Send [GET\_RESPONSE] on logical channel x with LE='XX' | <R\_APDU\_PART3>  SW=0x61XX |
| 4 | OCE → eUICC | Send [GET\_RESPONSE] on logical channel x with LE='XX' | <R\_APDU\_PART4>  SW=0x9000  The complete response is the result of the concatenation of all R-APDUs from <R\_APDU\_PART1> to <R\_APDU\_PART4> |

D.5 ES6 Requests And Responses

D.5.1 ES6 Requests

| Name | Content |
| --- | --- |
| METADATA\_WITH\_PPRS\_AND\_ICON | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileOwner {  mccMnc #MCC\_MNC1  },  profilePolicyRules {ppr1,ppr2}  } |
| METADATA\_WITH\_PPRS\_ICON\_AND\_NOTIF | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileOwner {  mccMnc #MCC\_MNC1  },  notificationConfigurationInfo {  { profileManagementOperation {  notificationInstall,  notificationLocalEnable,  notificationLocalDisable,  notificationLocalDelete  },  notificationAddress #TEST\_DP\_ADDRESS1  }  },  profilePolicyRules {ppr1,ppr2}  } |
| METADATA\_WITH\_PPRS\_ICON\_AND\_SPEC\_DATA | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileOwner {  mccMnc #MCC\_MNC1  },  serviceSpecificDataStoredInEuicc #VENDOR\_SPECIFIC\_EXTENSION1,  profilePolicyRules {ppr1,ppr2}  } |
| METADATA\_WITH\_PPRS\_ICON\_AND\_RPM\_CONFIG | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileOwner {  mccMnc #MCC\_MNC1  },  rpmConfiguration #RPM\_CONFIG\_OP\_PROF1,  profilePolicyRules {ppr1,ppr2}  } |
| METADATA\_WITH\_PPRS\_ICON\_AND\_HRI\_ADDR | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileOwner {  mccMnc #MCC\_MNC1  },  profilePolicyRules {ppr1,ppr2},  hriServerAddress #TEST\_HRI\_ADDRESS1  } |
| METADATA\_WITH\_PPRS\_ICON\_AND\_LPR\_CONFIG | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileOwner {  mccMnc #MCC\_MNC1  },  profilePolicyRules {ppr1,ppr2},  lprConfiguration {  pcmpAddress #TEST\_PCMP\_ADDRESS1  triggerLprOnEnableProfile  }  } |
| METADATA\_WITH\_PPRS\_ICON\_AND\_DC\_CONFIG\_DP | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileOwner {  mccMnc #MCC\_MNC1  },  profilePolicyRules {ppr1,ppr2},  deviceChangeConfiguration {  requestToDp {  smdpAddressToBeUsedForDc #TEST\_DP\_ADDRESS1  eidRequired  }  }  } |
| METADATA\_WITH\_PPRS\_ICON\_AND\_DC\_CONFIG\_AC | metadataReq StoreMetadataRequest ::= {  iccid #ICCID\_OP\_PROF1,  serviceProviderName #SP\_NAME1,  profileName #NAME\_OP\_PROF1,  iconType png,  icon #ICON\_OP\_PROF1,  profileOwner {  mccMnc #MCC\_MNC1  },  profilePolicyRules {ppr1,ppr2},  deviceChangeConfiguration {  usingStoredAc  {  activationCodeForDc ACTIVATION\_CODE\_1  deleteOldProfile  }  }  } |
| REMOVE\_PPR1 | metadataReq UpdateMetadataRequest ::= {  profilePolicyRules {ppr2}  } |
| UPD\_ENT\_CONFIG1 | metadataReq UpdateMetadataRequest ::= {  enterpriseConfiguration {  enterpriseOid #S\_ENTERPRISE\_OID,  enterpriseName #ENTERPRISE\_NAME1,  enterpriseRules {  enterpriseRuleBits {  referenceEnterpriseRule,  priorityEnterpriseProfile,  onlyEnterpriseProfilesCanBeInstalled  },  numberOfNonEnterpriseProfiles 0  }  } |
| UPD\_ICON\_REM\_PPR2 | metadataReq UpdateMetadataRequest ::= {  iconType jpg,  icon #ICON\_JPG,  profilePolicyRules {ppr1}  } |
| UPD\_NAMES\_REM\_PPRS\_V3 | metadataReq UpdateMetadataRequest ::= {  serviceProviderName #SP\_NAME2,  profileName #NAME\_OP\_PROF2,  tagsForDeletion '99'H  } |
| UPD\_NO\_METADATA | metadataReq UpdateMetadataRequest ::= { } |
| UPD\_PPR\_CONTROL | metadataReq UpdateMetadataRequest ::= {  serviceProviderName #SP\_NAME2,  profileName #NAME\_OP\_PROF2,  iconType jpg,  icon #ICON\_JPG,  profilePolicyRules {pprUpdateControl, ppr1}  } |
| UPD\_WITH\_EC | metadataReq UpdateMetadataRequest ::= {  serviceProviderName #SP\_NAME2,  enterpriseConfiguration {  enterpriseOid #S\_ENTERPRISE\_OID  enterpriseName #ENTERPRISE\_NAME1  }  } |
| UPD\_NAMES\_REM\_ICON\_REM\_PPRS\_V3 | metadataReq UpdateMetadataRequest ::= {  serviceProviderName #SP\_NAME2,  profileName #NAME\_OP\_PROF2,  tagsForDeletion { iconType, icon, {profilePolicyRules {ppr1, ppr2} }  } |
| UPD\_NOTIF\_CONFIG\_INFO | metadataReq UpdateMetadataRequest ::= {    notificationConfigurationInfo {  {  profileManagementOperation {  notificationLocalDelete  },  notificationAddress #TEST\_DP\_ADDRESS2  }  }  } |
| REM\_NOTIF\_CONFIG\_INFO | metadataReq UpdateMetadataRequest ::= {    tagsForDeletion {  'B6'  }  } |
| UPD\_SPEC\_DATA | metadataReq UpdateMetadataRequest ::= {    serviceSpecificDataStoredInEuicc #VENDOR\_SPECIFIC\_EXTENSION2  } |
| REM\_SPEC\_DATA | metadataReq UpdateMetadataRequest ::= {    tagsForDeletion 'BF22'H } |
| UPD\_RPM\_CONFIG | metadataReq UpdateMetadataRequest ::= {    rpmConfiguration {  managingDpList {},  profileOwnerOid #S\_PROFILE\_OWNER\_OID  }  } |
| REM\_ RPM\_CONFIG | metadataReq UpdateMetadataRequest ::= {    tagsForDeletion  'BA'H  } |
| UPD\_HRI\_ADDR | metadataReq UpdateMetadataRequest ::= {    hriServerAddress #TEST\_HRI\_ADDRESS3  } |
| REM\_HRI\_ADDR | metadataReq UpdateMetadataRequest ::= {    tagsForDeletion '9B'H } |
| UPD\_LPR\_CONFIG | metadataReq UpdateMetadataRequest ::= {    lprConfiguration {  pcmpAddress #TEST\_PCMP\_ADDRESS3  }  } |
| REM\_ LPR\_CONFIG | metadataReq UpdateMetadataRequest ::= {    tagsForDeletion 'BC'H } |
| UPD\_DC\_CONFIG\_DP | metadataReq UpdateMetadataRequest ::= {    deviceChangeConfiguration {  requestToDp {  smdpAddressToBeUsedForDc #TEST\_DP\_ADDRESS2  }  }  } |
| REM\_DC\_CONFIG\_DP | metadataReq UpdateMetadataRequest ::= {    tagsForDeletion 'BF20'H } |
| UPD\_DC\_CONFIG\_AC | metadataReq UpdateMetadataRequest ::= {    deviceChangeConfiguration {  usingStoredAc  {  activationCodeForDc ACTIVATION\_CODE\_2  }  }  } |
| REM\_DC\_CONFIG\_AC | metadataReq UpdateMetadataRequest ::= {    tagsForDeletion 'BF20'H } |
| REM\_SP\_NAME | metadataReq UpdateMetadataRequest ::= {    tagsForDeletion '91'H } |

D.6 VOID

D.7 VOID

D.8 VOID

D.9 VOID

Annex E Profiles

|  |  |
| --- | --- |
| Profile | GENERIC\_PROFILE\_STRUCTURE |
| Description | Generic Operational Profile ASN.1 structure to be used as a basis for all Profiles used in this specification. |
| Details | headerValue ProfileElement ::= header : {  major-version 2,  minor-version 3,  profileType "GSMA Profile Package",  iccid '89019990001234567893'H,  eUICC-Mandatory-services {  usim NULL,  milenage NULL  },  eUICC-Mandatory-GFSTEList {  -- see Note 1  id-MF,  id-USIM  }  }  mfValue ProfileElement ::= mf : {  mf-header {  mandated NULL,  identification 1  },  templateID id-MF,  mf {  fileDescriptor : {  pinStatusTemplateDO '01020A'H  }  },  ef-pl {  fileDescriptor : {  -- EF PL modified to use Access Rule 15 within EF ARR  securityAttributesReferenced '0F'H  }  },  ef-iccid {  -- swapped ICCID: 98109909002143658739  fillFileContent : '98109909002143658739'H  },  ef-dir {  fileDescriptor {  -- Shareable Linear Fixed File  -- 4 records, record length: 38 bytes  fileDescriptor '42210026'H,  efFileSize '98'H  },  -- USIM AID: A0000000871002FF33FF018900000100  fillFileContent  '61184F10A0000000871002FF33FF01890000010050045553494D'H  },  ef-arr {  fileDescriptor : {  fileDescriptor '42210025'H,  lcsi '05'H,  efFileSize '022B'H  },  fillFileContent : '8001019000800102A406830101950108800158A40683010A950108'H,  fillFileOffset : 10,  fillFileContent : '800101A40683010195010880015AA40683010A950108'H,  fillFileOffset : 15,  fillFileContent : '80015BA40683010A950108'H,  fillFileOffset : 26,  fillFileContent : '800101900080015A9700'H,  fillFileOffset : 27,  fillFileContent : '800103A406830101950108800158A40683010A950108'H,  fillFileOffset : 15,  fillFileContent : '800111A40683010195010880014AA40683010A950108'H,  fillFileOffset : 15,  fillFileContent : '800103A406830101950108800158A40683010A950108840132A406830101950108'H,  fillFileOffset : 4,  fillFileContent : '800101A406830101950108800102A406830181950108800158A40683010A950108'H,  fillFileOffset : 4,  fillFileContent : '800101900080011AA406830101950108800140A40683010A950108'H,  fillFileOffset : 10,  fillFileContent : '800101900080015AA40683010A950108'H,  fillFileOffset : 21,  fillFileContent : '8001019000800118A40683010A9501088001429700'H,  fillFileOffset : 16,  fillFileContent : '800101A40683010195010880015A9700'H,  fillFileOffset : 21,  fillFileContent : '800113A406830101950108800148A40683010A950108'H,  fillFileOffset : 15,  fillFileContent : '80015EA40683010A950108'H,  fillFileOffset : 26,  fillFileContent '8001019000800102A010A40683010195  0108A406830102950108800158A40683  010A950108'H  }  }  pukVal ProfileElement ::= pukCodes : {  puk-Header {  mandated NULL,  identification 2  },  pukCodes {  {  keyReference pukAppl1,  pukValue '3030303030303030'H,  -- maxNumOfAttemps:9, retryNumLeft:9  maxNumOfAttemps-retryNumLeft 153  },  {  keyReference pukAppl2,  pukValue '3132333435363738'H  },  {  keyReference secondPUKAppl1,  pukValue '3932393435363738'H,  -- maxNumOfAttemps:8, retryNumLeft:8  maxNumOfAttemps-retryNumLeft 136  }  }  }  pinVal ProfileElement ::= pinCodes : {  pin-Header {  mandated NULL,  identification 3  },  pinCodes pinconfig : {  {  keyReference pinAppl1,  pinValue '31323334FFFFFFFF'H,  unblockingPINReference pukAppl1  },  {  keyReference pinAppl2,  pinValue '30303030FFFFFFFF'H,  unblockingPINReference pukAppl2  },  {  keyReference adm1,  pinValue '35363738FFFFFFFF'H,  pinAttributes 1  }  }  }  usimValue ProfileElement ::= usim : {  usim-header {  mandated NULL,  identification 4  },  templateID id-USIM,  adf-usim {  fileDescriptor : {  fileID '7FF1'H,  dfName 'A0000000871002FF33FF018900000100'H,  pinStatusTemplateDO '01810A'H  }  },  ef-imsi {  -- numerical format: 234101943787656  fillFileContent '082943019134876765'H  },  ef-arr {  fileDescriptor {  linkPath '2F06'H  }  },  ef-ust {  -- Service Dialling Numbers, Short Message Storage…  fillFileContent '0A2E178CE73204000000000000'H  },  ef-spn {  -- ASCII format: "GSMA eUICC"  fillFileContent '0247534D41206555494343FFFFFFFFFFFF'H  },  ef-est {  -- Services deactivated  fillFileContent '00'H  },  ef-acc {  -- Access class 4  fillFileContent '0040'H  },  ef-ecc {  -- Emergency Call Code 911  fillFileContent '19F1FF01'H  }  }  usimPin ProfileElement ::= pinCodes : {  pin-Header {  mandated NULL,  identification 5  },  pinCodes pinconfig : {  {  keyReference secondPINAppl1,  pinValue '39323338FFFFFFFF'H  unblockingPINReference secondPUKAppl1,  -- PIN is Enabled  pinAttributes 1,  -- maxNumOfAttemps:2, retryNumLeft:2  maxNumOfAttemps-retryNumLeft 34  }  }  }  akaParamValue ProfileElement ::= akaParameter : {  aka-header {  mandated NULL,  identification 6  },  algoConfiguration algoParameter : {  algorithmID milenage,  -- RES and MAC 64 bits, CK and IK 128 bits  algorithmOptions '01'H,  key '000102030405060708090A0B0C0D0E0F'H,  opc '0102030405060708090A0B0C0D0E0F00'H,  -- rotationConstants uses default: '4000204060'H  -- xoringConstants uses default value  authCounterMax '010203'H  }  -- sqnOptions uses default: '02'H  -- sqnDelta uses default: '000010000000'H  -- sqnAgeLimit uses default: '000010000000'H  -- sqnInit uses default: all bytes zero  }  mnoSdValue ProfileElement ::= securityDomain : {  sd-Header {  mandated NULL,  identification 7  },  instance {  applicationLoadPackageAID 'A0000001515350'H,  classAID 'A000000151535041'H,  instanceAID 'A000000151000000'H,  applicationPrivileges '82FC80'H,  -- Secured  lifeCycleState '0F'H,  -- SCP80 supported  applicationSpecificParametersC9 '810280008201F08701F0'H,  -- other parameters MAY be necessary  applicationParameters {  -- TAR: B20100, MSL: 12  uiccToolkitApplicationSpecificParametersField  '0100000100000002011203B2010000'H  }  },  keyList {  {  -- C-ENC + R-ENC  keyUsageQualifier '38'H,  -- ENC key  keyIdentifier '01'H,  keyVersionNumber '01'H,  keyCompontents {  {  -- DES mode implicitly known (as an example)  keyType '80'H,  -- This value MAY be freely changed  keyData '112233445566778899AABBCCDDEEFF10'H  }  }  },  {  -- C-MAC + R-MAC  keyUsageQualifier '34'H,  -- MAC key  keyIdentifier '02'H,  keyVersionNumber '01'H,  keyCompontents {  {  -- DES mode implicitly known (as an example)  keyType '80'H,  -- This value MAY be freely changed  keyData '112233445566778899AABBCCDDEEFF10'H  }  }  },  {  -- C-DEK + R-DEK  keyUsageQualifier 'C8'H,  -- data ENC key  keyIdentifier '03'H,  keyVersionNumber '01'H,  keyCompontents {  {  -- DES mode implicitly known (as an example)  keyType '80'H,  -- This value MAY be freely changed  keyData '112233445566778899AABBCCDDEEFF10'H  }  }  },  -- AES Token Key (as an example)  -- This value MAY be freely changed  keyUsageQualifier '81'H,  -- MAY be used by SD  keyAccess '01'H,  -- Key Id 01  keyIdentifier '01'H,  keyVersionNumber '70'H,  keyCompontents {  {  -- AES (16 bytes key length)  -- This value MAY be freely changed  keyType '88'H,  -- This value MAY be freely changed  keyData 'CDFE56B7B72FAE6A047341F003D7A48D'H  }  }  },  {  -- Receipt (the AES scheme SHALL be supported)  keyUsageQualifier '44'H,  -- MAY be used by SD  keyAccess '01'H,  -- Key Id 01  keyIdentifier '01'H,  keyVersionNumber '71'H,  keyCompontents {  {  -- AES (16 bytes key length)  keyType '88'H,  -- This value MAY be freely changed  keyData '11121314212223243132333441424344'H  }  }  }  }  }  ssdValue ProfileElement ::= securityDomain : {  sd-Header {  mandated NULL,  identification 8  },  instance {  applicationLoadPackageAID 'A0000001515350'H,  classAID 'A000000151535041'H,  instanceAID 'A00000055910100102736456616C7565'H,  -- by default extradited under MNO-SD  -- Privileges: Security Domain + Trusted Path  applicationPrivileges '808000'H,  -- Personalized  lifeCycleState '0F'H,  -- SCP80 supported, extradiction supported  applicationSpecificParametersC9 '810280008201F0'H,  applicationParameters {  -- TAR: 6C7565, MSL: 12  uiccToolkitApplicationSpecificParametersField  '01000001000000020112036C756500'H  }  },  keyList {  {  -- C-ENC + R-ENC  keyUsageQualifier '38'H,  keyIdentifier '01'H,  keyVersionNumber '01'H,  keyCompontents {  {  -- DES mode implicitly known (as an example)  keyType '80'H,  -- This value MAY be freely changed  keyData '11223344556677881122334455667788'H  }  }  },  {  -- C-MAC + R-MAC  keyUsageQualifier '34'H,  -- MAC key  keyIdentifier '02'H,  keyVersionNumber '01'H,  keyCompontents {  {  -- DES mode implicitly known (as an example)  keyType '80'H,  -- This value MAY be freely changed  keyData '11223344556677881122334455667788'H  }  }  },  {  -- C-DEK + R-DEK  keyUsageQualifier 'C8'H,  -- data ENC key  keyIdentifier '03'H,  keyVersionNumber '01'H,  keyCompontents {  {  -- DES mode implicitly known (as an example)  keyType '80'H,  -- This value MAY be freely changed  keyData '11223344556677881122334455667788'H  }  }  }  }  }  rfmUicc ProfileElement ::= rfm : {  rfm-header {  identification 11  },  -- Instance AID  instanceAID ' A00000055910100001'H,  tarList {  'B00000'H  },  -- cryptographic checksum + counter higher  minimumSecurityLevel '12'H,  -- full access  uiccAccessDomain '00'H,  -- full access  uiccAdminAccessDomain '00'H  }  rfmUsim ProfileElement ::= rfm : {  rfm-header {  identification 12  },  -- Instance AID  instanceAID 'A00000055910100002'H,  tarList {  'B00020'H  },  -- cryptographic checksum + counter higher  minimumSecurityLevel '12'H,  -- full access  uiccAccessDomain '00'H,  -- full access  uiccAdminAccessDomain '00'H,  adfRFMAccess {  adfAID 'A0000000871002FF33FF018900000100'H,  -- UICC access condition: ADM1  adfAccessDomain '02000100'H,  -- UICC access condition: ADM1  adfAdminAccessDomain '02000100'H  }  }  endValue ProfileElement ::= end : {  end-header {  mandated NULL,  identification 99  }  } |
| *Note 1: The following OIDs are used:*  *id-MF OBJECT IDENTIFIER ::=*  *{joint-iso-itu-t(2) international-organizations(23) tca(143) euicc-profile(1) template(2) mf(1)}*  *id-USIM OBJECT IDENTIFIER ::=*  *{joint-iso-itu-t(2) international-organizations(23) tca(143) euicc-profile(1) template(2) usim(4)}* | |

|  |  |
| --- | --- |
| Profile | PROFILE\_OPERATIONAL1 |
| Description | Operational Profile  This Profile acts as an Operational Profile in the scope of this specification.  NOTE: Milenage algorithm is used in this Profile. |
| Details | The Profile Metadata SHALL be set to #METADATA\_OP\_PROF1, except if defined differently in the test sequence.  The Unprotected Profile Package content SHALL follow the ASN.1 structures specified above for GENERIC\_PROFILE\_STRUCTURE except that:   * the *iccid* field SHALL be set to #ICCID\_OP\_PROF1 in the *ProfileHeader* element, in non-swapped format * the ef-iccid present in the PE-MF SHALL be set to #ICCID\_OP\_PROF1 * the ef-imsi present in the PE-USIM SHALL be set to #IMSI\_OP\_PROF1 * the pinAttributes of pinAppl1 present in the PE\_PIN SHALL be set to 6 * the SCP80 encryption key configured in the PE-SecurityDomain that corresponds to the MNO-SD SHALL be set to #MNO\_SCP80\_ENC\_KEY * the SCP80 message authentication key configured in the PE-SecurityDomain that corresponds to the MNO-SD SHALL be set to #MNO\_SCP80\_AUTH\_KEY * the SCP80 data encryption key configured in the PE-SecurityDomain that corresponds to the MNO-SD SHALL be set to #MNO\_SCP80\_DATA\_ENC\_KEY * the instance AID configured in the PE-SecurityDomain that corresponds to the Supplementary Security Domain PE\_SSD SHALL be set to #SSD\_AID * the ef-dir present in the PE-MF SHALL be configured with the AID #USIM\_AID * the ef-ust SHALL be set in accordance to #EF\_UST1 (service 17 and 18 are not available) * the applicationPrivileges in PE-MNO-SD SHALL be set to '82DC00'H * the Token Verification and the Receipt Generation keys SHALL not be set in the PE-MNO-SD * the applicationSpecificParametersC9 in PE-MNO-SD SHALL be set to '810280008201F08701F0'H   The PROFILE\_OPERATIONAL1 UPP is named #UPP\_OP\_PROF1 in the scope of this document. |

|  |  |
| --- | --- |
| Profile | PROFILE\_OPERATIONAL2 |
| Description | Operational Profile  This Profile acts as an Operational Profile in the scope of this specification.  NOTE: Milenage algorithm is used in this Profile. |
| Details | The Profile Metadata SHALL be set to #METADATA\_OP\_PROF2, except if defined differently in the test sequence.  The Unprotected Profile Package content SHALL follow the ASN.1 structures specified above for GENERIC\_PROFILE\_STRUCTURE except that:   * the *iccid* field SHALL be set to #ICCID\_OP\_PROF2 in the *ProfileHeader* element, in non-swapped format * the ef-iccid present in the PE-MF SHALL be set to #ICCID\_OP\_PROF2 * the ef-imsi present in the PE-USIM SHALL be set to #IMSI\_OP\_PROF2 * The pinAttributes of pinAppl1 present in the PE\_PIN SHALL be set to 6 * the ef-ust SHALL be set in accordance to #EF\_UST1 (service 17 and 18 are not available) * the applicationPrivileges in PE-MNO-SD SHALL be set to '82DC00'H * the Token Verification and the Receipt Generation keys SHALL not be set in the PE-MNO-SD * the applicationSpecificParametersC9 in PE-MNO-SD SHALL be set to '810280008201F08701F0'H   The PROFILE\_OPERATIONAL2 UPP is named #UPP\_OP\_PROF2 in the scope of this document. |

|  |  |
| --- | --- |
| Profile | PROFILE\_OPERATIONAL3 |
| Description | Operational Profile with PPR2 but without notification  This Profile acts as an Operational Profile in the scope of this specification.  NOTE: Milenage algorithm is used in this Profile. |
| Details | The Profile Metadata SHALL be set to #METADATA\_OP\_PROF3, except if defined differently in the test sequence.  The Unprotected Profile Package content SHALL follow the ASN.1 structures specified above for GENERIC\_PROFILE\_STRUCTURE except that:   * the *iccid* field SHALL be set to #ICCID\_OP\_PROF3 in the *ProfileHeader* element, in non-swapped format * the ef-iccid present in the PE-MF SHALL be set to #ICCID\_OP\_PROF3 * the ef-imsi present in the PE-USIM SHALL be set to #IMSI\_OP\_PROF3 * the pinAttributes of pinAppl1 present in the PE\_PIN SHALL be set to 6 * the ef-ust SHALL be set in accordance to #EF\_UST1 (service 17 and 18 are not available) * the applicationPrivileges in PE-MNO-SD SHALL be set to '82DC00'H * the Token Verification and the Receipt Generation keys SHALL not be set in the PE-MNO-SD * the applicationSpecificParametersC9 in PE-MNO-SD SHALL be set to '810280008201F08701F0'H   The PROFILE\_OPERATIONAL3 UPP is named #UPP\_OP\_PROF3 in the scope of this document. |

|  |  |
| --- | --- |
| Profile | PROFILE\_OPERATIONAL4 |
| Description | Operational Profile with PPR1 and notification  This Profile acts as an Operational Profile in the scope of this specification.  NOTE: Milenage algorithm is used in this Profile. |
| Details | The Profile Metadata SHALL be set to #METADATA\_OP\_PROF4, except if defined differently in the test sequence.  The Profile Package content SHALL follow the ASN.1 structures specified above for GENERIC\_PROFILE\_STRUCTURE except that:   * the *iccid* field SHALL be set to #ICCID\_OP\_PROF4 in the *ProfileHeader* element, in non-swapped format * the ef-iccid present in the PE-MF SHALL be set to #ICCID\_OP\_PROF4 * the ef-imsi present in the PE-USIM SHALL be set to #IMSI\_OP\_PROF4 * the pinAttributes of pinAppl1 present in the PE\_PIN SHALL be set to 6 * the ef-ust SHALL be set in accordance to #EF\_UST1 (service 17 and 18 are not available) * the applicationPrivileges in PE-MNO-SD SHALL be set to '82DC00'H * the Token Verification and the Receipt Generation keys SHALL not be set in the PE-MNO-SD * the applicationSpecificParametersC9 in PE-MNO-SD SHALL be set to '810280008201F08701F0'H   The PROFILE\_OPERATIONAL4 UPP is named #UPP\_OP\_PROF4 in the scope of this document. |

|  |  |
| --- | --- |
| Profile | PROFILE\_OPERATIONAL5 |
| VOID (not used in this version of the specification) | |

|  |  |
| --- | --- |
| Profile | PROFILE\_OPERATIONAL6 |
| VOID (not used in this version of the specification) | |

|  |  |
| --- | --- |
| Profile | PROFILE\_OPERATIONAL7 |
| VOID (not used in this version of the specification) | |

|  |  |
| --- | --- |
| Profile | PROFILE\_OPERATIONAL8 |
| VOID (not used in this version of the specification) | |

|  |  |
| --- | --- |
| Profile | PROFILE\_OPERATIONAL9 |
| Description | Operational Profile with GID1 and GID2 set  This Profile acts as an Operational Profile in the scope of this specification.  NOTE: Milenage algorithm is used in this Profile. |
| Details | The Profile Metadata SHALL be set to #METADATA\_OP\_PROF9, except if defined differently in the test sequence.  The Unprotected Profile Package content SHALL follow the ASN.1 structures specified above for GENERIC\_PROFILE\_STRUCTURE except that:   * the *iccid* field SHALL be set to #ICCID\_OP\_PROF9 in the *ProfileHeader* element, in non-swapped format * the ef-iccid present in the PE-MF SHALL be set to #ICCID\_OP\_PROF9 * the ef-imsi present in the PE-USIM SHALL be set to #IMSI\_OP\_PROF9 * the pinAppl1 present in the PE\_PIN SHALL be enabled and has the value #PO1\_PIN1 * the ef-ust SHALL be set to #EF\_UST2 (service 17 and 18 are available) * the applicationPrivileges in PE-MNO-SD SHALL be set to '82DC00'H * the Token Verification and the Receipt Generation keys SHALL not be set in the PE-MNO-SD * the applicationSpecificParametersC9 in PE-MNO-SD SHALL be set to '810280008201F08701F0'H * the following new Profile Element PE\_OPT\_USIM SHALL be inserted right after PE\_USIM:  |  | | --- | | PE\_OPT\_USIM | | optusimValue ProfileElement ::= opt-usim : {  optusim-header {  mandated NULL,  identification 15  },  templateID id-OPT-USIM,  ef-gid1 {  fileDescriptor {  efFileSize '04'H  },  fillFileContent #GID1  },  ef-gid2 {  fileDescriptor {  efFileSize '04'H  },  fillFileContent #GID2  }  } | | NOTE: The following OIDs are used:  id-OPT-USIM OBJECT IDENTIFIER ::=  {joint-iso-itu-t(2) international-organizations(23) tca(143) euicc-profile(1) template(2) opt-usim(5)} |   The PROFILE\_OPERATIONAL9 UPP is named #UPP\_OP\_PROF9 in the scope of this document. |

|  |  |
| --- | --- |
| Profile | 5G\_PROFILE\_STRUCTURE |
| Description | 5G Operational Profile ASN.1 structure to be used as a basis for specific test scenario where explicitly stated. |
| Details | headerValue ProfileElement ::= header : {  major-version 3,  minor-version 1,  profileType "GSMA Profile Package",  iccid '89019990001234567893'H,  eUICC-Mandatory-services {  usim NULL,  milenage NULL,  get-identity NULL,  profile-a-x25519 NULL  },  eUICC-Mandatory-GFSTEList {  -- see Note 1  id-MF,  id-USIM  }  }  mfValue ProfileElement ::= mf : {  mf-header {  mandated NULL,  identification 1  },  templateID id-MF,  mf {  fileDescriptor : {  pinStatusTemplateDO '01020A'H  }  },  ef-pl {  fileDescriptor : {  -- EF PL modified to use Access Rule 15 within EF ARR  securityAttributesReferenced '0F'H  }  },  ef-iccid {  -- swapped ICCID: 98109909002143658739  fillFileContent : '98109909002143658739'H  },  ef-dir {  fileDescriptor : {  -- Shareable Linear Fixed File  -- 4 records, record length: 38 bytes  fileDescriptor '42210026'H,  efFileSize '98'H  },  -- USIM AID: A000000087100BFF33FF018900000100  fillFileContent :  '61184F10A000000087100BFF33FF01890000010050045553494D'H  },  ef-arr {  fileDescriptor : {  fileDescriptor '42210025'H,  lcsi '05'H,  efFileSize '022B'H  },  fillFileContent : '8001019000800102A406830101950108800158A40683010A950108'H,  fillFileOffset : 10,  fillFileContent : '800101A40683010195010880015AA40683010A950108'H,  fillFileOffset : 15,  fillFileContent : '80015BA40683010A950108'H,  fillFileOffset : 26,  fillFileContent : '800101900080015A9700'H,  fillFileOffset : 27,  fillFileContent : '800103A406830101950108800158A40683010A950108'H,  fillFileOffset : 15,  fillFileContent : '800111A40683010195010880014AA40683010A950108'H,  fillFileOffset : 15,  fillFileContent : '800103A406830101950108800158A40683010A950108840132A406830101950108'H,  fillFileOffset : 4,  fillFileContent : '800101A406830101950108800102A406830181950108800158A40683010A950108'H,  fillFileOffset : 4,  fillFileContent : '800101900080011AA406830101950108800140A40683010A950108'H,  fillFileOffset : 10,  fillFileContent : '800101900080015AA40683010A950108'H,  fillFileOffset : 21,  fillFileContent : '8001019000800118A40683010A9501088001429700'H,  fillFileOffset : 16,  fillFileContent : '800101A40683010195010880015A9700'H,  fillFileOffset : 21,  fillFileContent : '800113A406830101950108800148A40683010A950108'H,  fillFileOffset : 15,  fillFileContent : '80015EA40683010A950108'H,  fillFileOffset : 26,  fillFileContent : '8001019000800102A010A40683010195  0108A406830102950108800158A40683  010A950108'H  }  }  pukVal ProfileElement ::= pukCodes : {  puk-Header {  mandated NULL,  identification 2  },  pukCodes {  {  keyReference pukAppl1,  pukValue '3030303030303030'H,  -- maxNumOfAttemps:9, retryNumLeft:9  maxNumOfAttemps-retryNumLeft 153  },  {  keyReference pukAppl2,  pukValue '3132333435363738'H  },  {  keyReference secondPUKAppl1,  pukValue '3932393435363738'H,  -- maxNumOfAttemps:8, retryNumLeft:8  maxNumOfAttemps-retryNumLeft 136  }  }  }  pinVal ProfileElement ::= pinCodes : {  pin-Header {  mandated NULL,  identification 3  },  pinCodes pinconfig : {  {  keyReference pinAppl1,  pinValue '31323334FFFFFFFF'H,  unblockingPINReference pukAppl1  },  {  keyReference pinAppl2,  pinValue '30303030FFFFFFFF'H,  unblockingPINReference pukAppl2  },  {  keyReference adm1,  pinValue '35363738FFFFFFFF'H,  pinAttributes 1  }  }  }  usimValue ProfileElement ::= usim : {  usim-header {  mandated NULL,  identification 4  },  templateID id-USIM,  adf-usim {  fileDescriptor : {  fileID '7FF1'H,  dfName 'A000000087100BFF33FF018900000100'H,  pinStatusTemplateDO '01810A'H  }  },  ef-imsi {  doNotCreate : NULL  },  ef-arr {  fileDescriptor : {  linkPath '2F06'H  }  },  ef-ust {  -- Service Dialling Numbers, Short Message Storage…  fillFileContent : '0A2E178CE7320400000000000000001882'H  },  ef-spn {  -- ASCII format: "GSMA eUICC"  fillFileContent : '0247534D41206555494343FFFFFFFFFFFF'H  },  ef-est {  -- Services deactivated  fillFileContent : '00'H  },  ef-acc {  -- Access class 4  fillFileContent : '0040'H  },  ef-ecc {  -- Emergency Call Code 911  fillFileContent : '19F1FF01'H  }  }  usimPin ProfileElement ::= pinCodes : {  pin-Header {  mandated NULL,  identification 5  },  pinCodes pinconfig : {  {  keyReference secondPINAppl1,  pinValue '39323338FFFFFFFF'H,  unblockingPINReference secondPUKAppl1,  -- PIN is Enabled  pinAttributes 1,  -- maxNumOfAttemps:2, retryNumLeft:2  maxNumOfAttemps-retryNumLeft 34  }  }  }  df5gsValue ProfileElement ::= df-5gs : {  df-5gs-header {  mandated NULL,  identification 556  },  templateID { 2 23 143 1 2 13 3 },  df-df-5gs {  fileDescriptor : {  pinStatusTemplateDO '01810A'H  }  },  ef-supinai {  fileDescriptor : {  lcsi '05'H,  efFileSize '18'H  },  fillFileContent : '801174657374757365724067736D612E6F7267'H  },  ef-routing-indicator {  }  }  dfsaipValue ProfileElement ::= df-saip : {  df-saip-header {  mandated NULL,  identification 261  },  templateID { 2 23 143 1 2 14 },  df-df-saip {  fileDescriptor : {  lcsi '05'H,  pinStatusTemplateDO '01810A'H  }  },  ef-suci-calc-info-usim {  fileDescriptor : {  lcsi '05'H,  efFileSize '04'H,  proprietaryEFInfo {  specialFileInformation '40'H,  fileDetails '01'H  }  },  fillFileContent : 'A0020100'H  }  }  akaParamValue ProfileElement ::= akaParameter : {  aka-header {  mandated NULL,  identification 6  },  algoConfiguration algoParameter : {  algorithmID milenage,  -- RES and MAC 64 bits, CK and IK 128 bits  algorithmOptions '01'H,  key '000102030405060708090A0B0C0D0E0F'H,  opc '0102030405060708090A0B0C0D0E0F00'H,  -- rotationConstants uses default: '4000204060'H  -- xoringConstants uses default value  authCounterMax '010203'H  }  -- sqnOptions uses default: '02'H  -- sqnDelta uses default: '000010000000'H  -- sqnAgeLimit uses default: '000010000000'H  -- sqnInit uses default: all bytes zero  }  mnoSdValue ProfileElement ::= securityDomain : {  sd-Header {  mandated NULL,  identification 7  },  instance {  applicationLoadPackageAID 'A0000001515350'H,  classAID 'A000000151535041'H,  instanceAID 'A000000151000000'H,  applicationPrivileges '82FC80'H,  -- Secured  lifeCycleState '0F'H,  -- SCP80 supported  applicationSpecificParametersC9 '810280008201F08701F0'H,  -- other parameters MAY be necessary  applicationParameters {  -- TAR: B20100, MSL: 12  uiccToolkitApplicationSpecificParametersField  '0100000100000002011203B2010000'H  }  },  keyList {  {  -- C-ENC + R-ENC  keyUsageQualifier '38'H,  -- ENC key  keyIdentifier '01'H,  keyVersionNumber '01'H,  keyCompontents {  {  -- DES mode implicitly known (as an example)  keyType '80'H,  -- This value MAY be freely changed  keyData '112233445566778899AABBCCDDEEFF10'H  }  }  },  {  -- C-MAC + R-MAC  keyUsageQualifier '34'H,  -- MAC key  keyIdentifier '02'H,  keyVersionNumber '01'H,  keyCompontents {  {  -- DES mode implicitly known (as an example)  keyType '80'H,  -- This value MAY be freely changed  keyData '112233445566778899AABBCCDDEEFF10'H  }  }  },  {  -- C-DEK + R-DEK  keyUsageQualifier 'C8'H,  -- data ENC key  keyIdentifier '03'H,  keyVersionNumber '01'H,  keyCompontents {  {  -- DES mode implicitly known (as an example)  keyType '80'H,  -- This value MAY be freely changed  keyData '112233445566778899AABBCCDDEEFF10'H  }  }  },  {  -- AES Token Key (as an example)  -- This value MAY be freely changed  keyUsageQualifier '81'H,  -- MAY be used by SD  keyAccess '01'H,  -- Key Id 01  keyIdentifier '01'H,  keyVersionNumber '70'H,  keyCompontents {  {  -- AES (16 bytes key length)  -- This value MAY be freely changed  keyType '88'H,  -- This value MAY be freely changed  keyData 'CDFE56B7B72FAE6A047341F003D7A48D'H  }  }  },  {  -- Receipt (the AES scheme SHALL be supported)  keyUsageQualifier '44'H,  -- MAY be used by SD  keyAccess '01'H,  -- Key Id 01  keyIdentifier '01'H,  keyVersionNumber '71'H,  keyCompontents {  {  -- AES (16 bytes key length)  keyType '88'H,  -- This value MAY be freely changed  keyData '11121314212223243132333441424344'H  }  }  }  }  }  ssdValue ProfileElement ::= securityDomain : {  sd-Header {  mandated NULL,  identification 8  },  instance {  applicationLoadPackageAID 'A0000001515350'H,  classAID 'A000000151535041'H,  instanceAID 'A00000055910100102736456616C7565'H,  -- by default extradited under MNO-SD  -- Privileges: Security Domain + Trusted Path  applicationPrivileges '808000'H,  -- Personalized  lifeCycleState '0F'H,  -- SCP80 supported, extradiction supported  applicationSpecificParametersC9 '810280008201F0'H,  applicationParameters {  -- TAR: 6C7565, MSL: 12  uiccToolkitApplicationSpecificParametersField  '01000001000000020112036C756500'H  }  },  keyList {  {  -- C-ENC + R-ENC  keyUsageQualifier '38'H,  keyIdentifier '01'H,  keyVersionNumber '01'H,  keyCompontents {  {  -- DES mode implicitly known (as an example)  keyType '80'H,  -- This value MAY be freely changed  keyData '11223344556677881122334455667788'H  }  }  },  {  -- C-MAC + R-MAC  keyUsageQualifier '34'H,  -- MAC key  keyIdentifier '02'H,  keyVersionNumber '01'H,  keyCompontents {  {  -- DES mode implicitly known (as an example)  keyType '80'H,  -- This value MAY be freely changed  keyData '11223344556677881122334455667788'H  }  }  },  {  -- C-DEK + R-DEK  keyUsageQualifier 'C8'H,  -- data ENC key  keyIdentifier '03'H,  keyVersionNumber '01'H,  keyCompontents {  {  -- DES mode implicitly known (as an example)  keyType '80'H,  -- This value MAY be freely changed  keyData '11223344556677881122334455667788'H  }  }  }  }  }  rfmUicc ProfileElement ::= rfm : {  rfm-header {  identification 11  },  -- Instance AID  instanceAID ' A00000055910100001'H,  tarList {  'B00000'H  },  -- cryptographic checksum + counter higher  minimumSecurityLevel '12'H,  -- full access  uiccAccessDomain '00'H,  -- full access  uiccAdminAccessDomain '00'H  }  rfmUsim ProfileElement ::= rfm : {  rfm-header {  identification 12  },  -- Instance AID  instanceAID 'A00000055910100002'H,  tarList {  'B00020'H  },  -- cryptographic checksum + counter higher  minimumSecurityLevel '12'H,  -- full access  uiccAccessDomain '00'H,  -- full access  uiccAdminAccessDomain '00'H,  adfRFMAccess {  adfAID ' A000000087100BFF33FF018900000100'H,  -- UICC access condition: ADM1  adfAccessDomain '02000100'H,  -- UICC access condition: ADM1  adfAdminAccessDomain '02000100'H  }  }  endValue ProfileElement ::= end : {  end-header {  mandated NULL,  identification 99  }  } |
| *Note 1: The following OIDs are used:*  *id-MF OBJECT IDENTIFIER ::=*  *{joint-iso-itu-t(2) international-organizations(23) simalliance(143) euicc-profile(1) template(2) mf(1)}*  *id-USIM OBJECT IDENTIFIER ::=*  *{joint-iso-itu-t(2) international-organizations(23) simalliance(143) euicc-profile(1) template(2) usim(4) version2(2)}* | |

|  |  |
| --- | --- |
| Profile | PROFILE\_OPERATIONAL10 |
| Description | Operational Profile  This Profile acts as an Operational Profile in the scope of this specification.  NOTE: Milenage algorithm is used in this Profile |
| Details | The Profile Metadata SHALL be set to #METADATA\_OP\_PROF10, except if defined differently in the test sequence.  The Unprotected Profile Package content SHALL follow the ASN.1 structure specified above for 5G\_PROFILE\_STRUCTURE except that:   the *iccid* field SHALL be set to #ICCID\_OP\_PROF10 in the *ProfileHeader* element, in non-swapped format   the ef-iccid present in the PE-MF SHALL be set to #ICCID\_OP\_PROF10   the pinAttributes of pinAppl1 present in the PE\_PIN SHALL be set to 6   the SCP80 encryption key configured in the PE-SecurityDomain that corresponds to the MNO-SD SHALL be set to #MNO\_SCP80\_ENC\_KEY   the SCP80 message authentication key configured in the PE-SecurityDomain that corresponds to the MNO-SD SHALL be set to #MNO\_SCP80\_AUTH\_KEY   the SCP80 data encryption key configured in the PE-SecurityDomain that corresponds to the MNO-SD SHALL be set to #MNO\_SCP80\_DATA\_ENC\_KEY   the instance AID configured in the PE-SecurityDomain that corresponds to the Supplementary Security Domain PE\_SSD SHALL be set to #SSD\_AID   the ef-dir present in the PE-MF SHALL be configured with the AID #USIM\_AID   the applicationPrivileges in PE-MNO-SD SHALL be set to '82DC00'H   * the Token Verification and the Receipt Generation keys SHALL not be set in the PE-MNO-SD    the applicationSpecificParametersC9 in PE-MNO-SD SHALL be set to '810280008201F08701F0'H |

Annex F eUICC Settings

F.1 eUICC Settings

In order to execute the test cases defined in this document, the eUICC Manufacturer SHALL deliver following settings:

|  |  |
| --- | --- |
| eUICC Setting name | Description |
| IUT\_ADDITIONAL\_EUICC\_INFO | Information about the eUICC as defined by the EUM. It MAY correlate with the additional issuer information contained in the EID |
| IUT\_DLOA\_URL | Discovery Base URL of the SE default DLOA Registrar as defined in GlobalPlatform DLOA specification [19] (optional). |
| IUT\_EUICC\_ADD\_PP\_VERSIONS | The expected content, if any, of the additionalEuiccProfilePackageVersions field in EUICCInfo2, coded as binary value without tag and length.  The version(s) indicated in this field SHALL be version(s) listed in section 7.1 in the “Allowed values for #IUT\_EUICC\_ADD\_PP\_VERSIONS” column. |
| IUT\_EUICC\_CATEGORY | The category SHALL be present as defined below:   * other(0) * or basicEuicc(1) * or mediumEuicc(2) * or contactlessEuicc(3) |
| IUT\_EUICC\_CERT\_CHAIN\_VARIANT | Preferred eUICC Certificate chain variant. It SHALL be either O, Ov3, A, B or C.  NOTE: the eUICC MAY be configured with several Certificates using different Variants. Unless otherwise specified, the test sequences in this document let the eUICC use the variant indicated in this IUT setting. |
| IUT\_EUICC\_CERT\_CHAIN\_VARIANT \_CI\_PK\_ID | The Root CI Public Key Identifier of the Root CI corresponding to the eUICC Certificate implied by #IUT\_EUICC\_CERT\_CHAIN\_VARIANT. It SHALL be a Root CI defined in SGP.26 [25]. |
| IUT\_EUICC\_FF\_TYPE | The form factor type of the eUICC as an ASN.1 INTEGER value, either removableEuicc(0) if eUICC can be removed, or nonRemovableEuicc(1) if eUICC cannot be removed- |
| IUT\_EUICC\_FIRMWARE\_VER | eUICC Firmware version coded as binary value (3 bytes representing major/minor/revision). |
| IUT\_EUICC\_MULTIPLEXING\_LSI\_INDICATION | One of:   * MANAGE LSI(Select LSI) * T=1 + NAD byte |
| IUT\_GLOBALPLATFORM\_VERSION | GlobalPlatform version coded as binary value (3 bytes representing major/minor/revision, 2.3.0 or higher). The support of GlobalPlatform is considered as mandatory in the scope of this specification. |
| IUT\_MEP\_LSI\_OPTIONS | LSI options supported by the eUICC and Terminal |
| IUT\_MEP\_MAX\_LSIS | Maximum number of LSIs supported for Enabled Profiles by the eUICC |
| IUT\_PLATFORM\_LABEL | Platform\_Label as defined in GlobalPlatform DLOA specification [19] (optional). |
| IUT\_PP\_VERSION | Protection Profile version coded as binary value (3 bytes representing major/minor/revision). |
| IUT\_RSP\_VERSION\_HIGHEST | Highest SGP.22 version supported, encoded as the value part of an ASN.1 VersionType (e.g. 0x03 01 00) |
| IUT\_SAS\_ACCREDITATION\_NUMBER | SAS Accreditation Number, coded as ASN.1 UTF8String. |
| IUT\_SERVER\_CI\_PK\_ID | The Root CI Public Key Identifier of a Root CI on the curve of which the eUICC supports server Certificates chain verification.  It SHALL be a Root CI defined in SGP.26 [25]. |
| IUT\_SIMA\_VERSION | Version of TCA (formerly SIMalliance) eUICC Profile Package Specification [4] supported by the eUICC (3 bytes representing major/minor/revision) e.g. 0x020100. |
| IUT\_TS102241\_VERSION | The ts102241 version field is coded as binary value (3 bytes representing major/minor/revision, 17.0.0 or higher).  The ts102241 Version field indicates the latest version of ETSI TS102 241[17] supported by the eUICC. |
| IUT\_UICC\_CAPABILITY | Sequence is derived from ServicesList[] defined in eUICC Profile Package PEDefinitions, coded as ASN.1 BIT STRING (19 bits). |

F.2 VOID

F.3 VOID

F.4 VOID

Annex G Initial States

Unless it is defined differently in a particular test case, the IUTs SHALL be set in the following initial state before the test case execution.

G.1 VOID

G.2 eUICC

Depending on the test cases and on the supported options, the EUM SHALL configure the eUICC according to the following Initial States. The initial conditions SHALL be restored, as described in the following subsections, after each test sequence.

G.2.1 Common Initial States

The following initial states apply for all test cases defined in this Test Plan whatever the options supported by the eUICC:

 The eUICC is configured with the ISD-R AID #ISD\_R\_AID and the EID #EID1.

 The eUICC does not contain any Profile.

 The eUICC’s Pending Notifications List is empty.

 No RSP session is ongoing.

 The eUICC is configured with the default SM-DS address #TEST\_ROOT\_DS\_ADDRESS.

* No additional Root DS addresses has been configured.

 The eUICC is configured without Default SM-DP+ address.

 No CRL is loaded on the eUICC.

 The ECASD is configured as defined in section G.2.2 and/or G.2.3 depending on the support of the options O\_E\_NIST and O\_E\_BRP. Other configurations are FFS.

o If the eUICC only supports O\_E\_NIST, the ECASD is configured as defined in section G.2.2.

o If the eUICC only supports O\_E\_BRP, the ECASD is configured as defined in section G.2.3.

o If the eUICC supports O\_E\_NIST and O\_E\_BRP, the ECASD is configured as defined in sections G.2.2 and G.2.3 (i.e. several EUM / eUICC Certificates and Keys are configured in the eUICC).

 Independent of configurations defined in sections G.2.2 and/or G.2.3, the eUICC may be configured with certificates according to the different variants identified by the following options: O\_varO, O\_varOv3, O\_varA, O\_varB, O\_varC.

The CI, identified as highest priority in euiccCiPKIdListForSigning or (if variants A/B/C are supported) in euiccCiPKIdListForSigningV3, is also selectable in the euiccCiPKIdListForVerification (i.e. all EUM and eUICC Certificates lead to a Root CI certificate linked to a #PK\_CI\_SIG contained in the eUICC).

This CI corresponds to the SubjectKeyIdentifier of one of the #CERT\_CI\_SIG defined in sections G.2.2 and G.2.3.

The RAT configuration defined in section G.2.4 applies for all test sequences if Multiple Enabled Profiles feature is not supported by the eUICC, except if the Test Case overrides it. Alternative RAT configurations for specific Test Cases or eUICC settings are defined in sections G.2.5, G.2.6 and G.2.7.

G.2.2 For eUICC supporting NIST P-256

If the eUICC supports O\_E\_NIST, the ECASD contains at least:

 The eUICC’s Private Key #SK\_EUICC\_SIG (for creating ECDSA signatures) based on NIST P-256 [11]

 The eUICC’s Certificate #CERT\_EUICC\_SIG (for eUICC authentication) containing the eUICC’s Public Key #PK\_EUICC\_SIG based on NIST P-256 [11]

 A Root Certificate Issuer’s Public Key #PK\_CI\_SIG (for verifying off-card entities certificates) defined in SGP.26 [25] and based on NIST P-256 [11]

 The Certificate of the EUM #CERT\_EUM\_SIG defined in SGP.26 [25] and based on NIST P-256 [11]

Other Certificates and Keys MAY be present.

G.2.3 For eUICC supporting BrainpoolP256r1

If the eUICC supports O\_E\_BRP, the ECASD contains at least:

 The eUICC’s Private Key #SK\_EUICC\_SIG (for creating ECDSA signatures) based on brainpoolP256r1 [8]

 The eUICC’s Certificate #CERT\_EUICC\_SIG (for eUICC authentication) containing the eUICC’s Public Key #PK\_EUICC\_SIG based on brainpoolP256r1 [8]

 A Root Certificate Issuer’s Public Key #PK\_CI\_SIG (for verifying off-card entities certificates) defined in SGP.26 [25] and based on brainpoolP256r1 [8]

 The Certificate of the EUM #CERT\_EUM\_SIG defined in SGP.26 [25] and based on brainpoolP256r1 [8]

 Other Certificates and Keys MAY be present.

G.2.4 With RAT configuration for SEP

The eUICC’s RAT is configured as detailed in SGP.22 section 2.9.2:

 Only one PPAR authorizing PPR1 and PPR2 for all MNOs with End User consent required i.e. #PPRS\_ALLOWED

This configuration SHALL be used as a default if the eUICC does not support MEP feature.

G.2.5 With Additional PPARs in the RAT

The eUICC’s RAT is configured as below (following this order):

 Additional PPARs representing custom agreements between MNOs and OEMs:

o #PPR1\_WITH\_OWNER\_GID

o #PPR1\_WITHOUT\_GID

o #PPR2\_WITHOUT\_CONSENT

 The last PPAR authorizes PPR1 and PPR2 for all MNOs with End User consent required i.e. #PPRS\_ALLOWED

This configuration is used in specific test cases if the eUICC does not support MEP feature.

G.2.6 Clean-up procedure

Unless differently specified in the test case, the following procedure SHALL be executed after each test sequence to bring the eUICC back to its Common Initial State:

* eUICC Memory Reset to delete all profiles and reset the SM-DP+ Address
* Retrieve and Remove all pending notifications

Where necessary, in addition to the above, other steps may be executed to restore the initial state specified in this Annex.

G.2.7 With RAT configuration for MEP

The eUICC’s RAT is configured as below:

 Only one PPAR authorizing PPR2 for all MNOs with End User consent required i.e. #PPR2\_ALLOWED

This configuration SHALL be used as a default if the eUICC supports MEP feature and allows PPR2.

G.3 VOID

Annex H Icons and QR Codes

The files for the eUICC Consumer Devices Icons are provided within SGP.23\_AnnexH\_Icons.zip package, which accompany the present document.

Annex J Integrated eUICC Testing (Normative)

J.1 Overview (Informative)

An Integrated eUICC hardware resides in an SoC along with other subsystems such as general processing and mobile broadband modem, all connected through a proprietary SoC interconnect channel. Alternatively, an Integrated eUICC may communicate with a mobile broadband modem external to the SoC via an external interface, which may be proprietary or based on a standard not associated with UICC. As such, Integrated eUICC may not include a physical UICC-Terminal interface [5].

In order to test the functionality and compliancy of an Integrated eUICC, hardware and OS, Integrated eUICC manufacturers need to provide and support a test interface to which testing equipment can be connected to. Having a standardized testing interface, will increase interoperable and reusability between different manufacturer of Integrated eUICC and test equipment.

For Integrated eUICC with a USB CCID [28] test interface, this annex describes its properties. In cases where a USB interface is not available in a device containing an Integrated eUICC, an adapter to USB CCID needs to be provided, e.g. Bluetooth to USB CCID. The functionality needed to provide and support the test interface, shall be considered part of the test environment and not the IUT.



Integrated eUICC with USB CCID [28] Test Interface

Note: The mechanism providing USB CCID to the RSP eUICC Test System, and described in this Annex, is implementation specific. As such, it may be implemented in the SoC, on-Device, off-Device or any combination thereof.

J.2 Integrated eUICC test requirements

An Integrated eUICC manufacturer shall provide a USB CCID test interface implementing the functionality specified in J.3.

The test interface shall maintain the integrity and order of the data between the Integrated eUICC and the test system.

The Integrated eUICC manufacturer shall ensure that during testing no other clients or SoC subsystems interfere with the testing.

The Integrated eUICC may use any physical or logical interface between the Integrated eUICC and the test system, as long as a USB CCID is provided to the test system and the channel is reliable (i.e. maintain integrity and order).

J.3 USB CCID test interface

The Integrated eUICC USB CCID test interface shall operate in a card reader mode.

The Integrated eUICC USB CCID test interface shall support the following [26] section 6 messages:

* [26] section 6.1 Messages:
  + PC\_to\_RDR\_IccPowerOn
  + PC\_to\_RDR\_IccPowerOff
  + PC\_to\_RDR\_GetSlotStatus
  + PC\_to\_RDR\_Escape
  + PC\_to\_RDR\_XfrBlock
  + PC\_to\_RDR\_T0APDU
  + PC\_to\_RDR\_Secure
  + PC\_to\_RDR\_Abort
* [26] section 6.2 Messages:
  + RDR\_to\_PC\_SlotStatus
  + RDR\_to\_PC\_Escape
  + RDR\_to\_PC\_DataBlock

Note: For test systems using wincard.h/PCSC lite APIs to connect to the Integrated eUICC USB CCID test interface, the following APIs are expected to be used:

* SCardEstablishContext
* SCardListReaders[A|W]
* SCardConnect[A|W]
* SCardControl
* SCardTransmit
* SCardDisconnect
* SCardStatus[A|W]
* SCardReleaseContext
* SCardReconnect
* SCardBeginTransaction
* SCardEndTransaction
* SCardGetStatusChange
* SCardFreeMemory
* SCardGetAttrib

Annex L Document Management

L.1 Document History

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Version | Date | CR No | Brief Description of Change | Entity | Approval Authority | Editor / Company |
| v1.0 | 9th June 2017 |  | Initial version of SGP.23 v1.0 Test Specification |  | PSMC | Yolanda Sanz, GSMA |
| v1.1 | 28th Sept 2017 |  | Minor version of SGP.23 Test specifications |  | RSPLEN | Yolanda Sanz, GSMA |
| v1.2 | 3rd Jan 2018 |  | Minor version of SGP.23 Test specifications |  | RSPLEN | Yolanda Sanz, GSMA |
| SGP.23-1 V2.0 Draft 0 | 27th April 2018 |  | The first draft 1 after Dividing SGP.23 in three different documents (SGP.23-1, 2 and 3) |  | RSPTEST | Sebastien Kuras, FIME |
| SGP.23-1 v3.1 Draft 31 | 5 October 2023 |  | Draft sent to eSIM for issuing | eUICC | eSIMWG3 | Guido Abate, STMicroelectronics |
| SGP.23-1 v3.1.1 Draft 0 | 17 October 2023 | Editor’s review | Same content as SGP.23-1 v3.1 Draft 31. All changes accepted | eUICC | ISAG | Guido Abate, STMicroelectronics |
| SGP.23-1 v3.1.1 Draft 1 | 31 October 2023 | CR311003r01 | MEP\_MEP-B without REFRESH optional | eUICC | Guido Abate, STMicroelectronics |
| CR311004R01 | ES10c\_EnableProfile\_MEPA1\_A2 |
| CR311007 | FixContactPcmpErrorCodes |
| CR311008R01 | MultipleRPMCommands |
| CR311009 | FixStoreMetadataEnterpriseProfiles |
| CR311011 | RemovePPRs |
| Editor’s review | As agreed during eSIMWG3#99:   * Modified condition in applicability table for test 4.2.27.2.1, sequences 1 and 2: from 'M' to 'C319' * Modified condition in applicability table for test 5.2.3.2.1: from 'M' to 'C319' * Voided section G.2.7 |
| SGP.23-1 v3.1.1 Draft 2 | 5 December 2023 | Editor’s review | As agreed during eSIMWG3#100:   * Un-voided section G.2.7 | eUICC | Guido Abate, STMicroelectronics |
| CR311002R04 | MEP\_Fix\_eUICCMemoryReset\_GetProfileInfo\_TstSeq |
| Editor’s review | Fixed tables shading in the MEP test sequences |
| CR311005R01 | ES10c\_DisableProfile\_MEPA1\_A2 |
| CR311006R01 | Fix\_MEP\_ApplicabilityTable\_UpdateMetaDataV3Param |
| CR311012 | FixRpmDeleteProfile |
| CR311014 | FixCodingOfEnterpriseRules |
| CR311015R01 | FixRpmUpdateMetadata |
| CR311016R01 | FixRpmEnterpriseProfiles |
| CR311017 | FixMccMnc3 |
| CR311001r04 | MEP\_Fix MEP-B Test Sequences |
| CR311013R02 | FixRpmListProfileInfo |
| CR311020 | MetadataOpProf10 |
| CR311021 | AddMissingCommands |
| CR311022R01 | FixRpmMultipleCommands |
| CR311023 | FixAuthServer |
| CR311024 | MissingConstants |
| CR311025 | FixUpdateMetadata |
| CR311026 | FixRpmMetadata |
| CR311028r00 | Specify LSI multiplexing |
| CR311010 | UpdateMetadataEnterpriseConfig |
| CR311031r01 | Reduce cognitive load - What's a procedure |
| SGP.23-1 v3.1.1 Draft 3 | 20 December 2023 | CR311029r00 | Clarify management of Logical Channels | eUICC | Guido Abate, STMicroelectronics |
| CR311032r03 | Retrofit CR311001 on MEP-B Test Sequences for Disable |
| CR311034R00 | Refactoring of procedural methods is FFS |
| CR311035R03 | Retrofit CR311001 on Error Test Sequences |
| CR311036R03 | Additional fixes MEP tests |
| CR311037R01 | Revising Applicability Table regarding PPR1 in MEP |
| Editor’s review | Reviewed Applicability table for consistency:   * “test sequence” was sometimes written with initials uppercase -> all in lowercase now * “sequence” was often singular in spite it indicated more than one * “sequence” and “test sequence” were both used. Now “test sequence” is consistently used * “test sequence” is sometimes preceded by “the”, some other times it is not. The article is now consistently used. |
| CR311038 | Fix\_MTD\_RES\_RPR\_FOR\_SINGLE\_CMND |
| CR311039 | PoR\_for\_UpdateMetadata |
| CR311040 | Select\_ISDR |
|  |  | Editor’s review | * Removed empty lines from tables in annexes * Removed a duplicated sentence in section 2.2.3.5 * Fixed an implementation error of CR2343R01 in section 2.2.3.4 * Fixed font sizes in several parts of the document |
| SGP.23-1 v3.1.2 | 26 April 2024 | Editor’s review | Same content as SGP.23-1 v3.1.1 Draft 3. All changes accepted | eUICC | Guido Abate, STMicroelectronics |
| a | Revising Applicability Table regarding PPR1 in MEP  (Note: in v3.1.1 Draft 3 R01 was implemented) | eUICC | Guido Abate, STMicroelectronics |
| Editor’s review | Fixed an implementation error of CR311040 “Select\_ISDR” in section 4.2.1.2.1 |
| CR312003r00 | Typo in Refresh mode 'UICC Reset' |
| CR312004r01 | Clarify definition of IUT\_MEP\_MODE |
| Editor’s review | Changed first page as per new GSMA template | eUICC | Guido Abate, STMicroelectronics |
| CR312002R02 | Fix\_MTD\_MEP\_TERMINAL\_PROFILE |
| CR312008R00 | RevertRpmTestMetadata |
| CR312001R07 | Reduce Cognitive Load - Fix MEP Procedures Methods Constants |
| CR312005R01 | Adjust applicability table |
| CR312011R01 | Align ATR+ISDR Selection with methods and procedures |
| CR312013R00 | Cleanup target port from MEP-A2 command |
|  |  |
| CR312014R00 | Fix empty command in MEP testing |
| CR312007R00 | Java Card Optionality SGP23-1 eUICC v3.1.2 | eUICC | eSIMWG3 | Guido Abate, STMicroelectronics |
| CR312009r02 | Align disableProfile with methods and procedures |
| CR312012r01 | Align eUICCMemoryReset with methods and procedures |
| CR312016R01 | Fix leftovers of method renaming | eUICC | eSIMWG3 | Guido Abate, STMicroelectronics |
| CR312017R04 | Fix\_AuthServerTestSequenceHeader |
| CR312018R00 | Make\_MEP\_TestSequence\_FlexibleFor\_T0\_T1 |
| CR312021R02 | MEP Cleanup |
| CR312022r03 | Support Variants of server Certificates chains |
| CR312023R01 | Fix\_MEP-Flexisibility\_EnableProfile |
| CR312024R00 | Fix\_MEP-Flexisibility\_DisableProfile |
| CR312025R00 | Fix\_MEP-Flexisibility\_eUICCMemoryReset |
| CR312026R00 | Fix\_MEP-Flexisibility\_GetProfileInfo\_ATR |
| CR312027r00 | Fixes\_in\_AnnexE |
| CR312028R00 | Remove O\_E\_SERVICE\_SPECIFIC\_DATA\_IN\_PM |
| CR3120029R01 | Fix Variant OV3 on the conditions |
| CR3120030R00 | Fix condtions on applicability table |
| Editor’s review | Fixed #IUT\_RSP\_VERSION\_HIGHEST that is now set to v3.1 as agreed during eSIMWG3#107 |
| CR312020R00 | Fix\_GetEuiccConfiguredDataResponse |

L.2 Other Information

|  |  |
| --- | --- |
| Type | Description |
| Document Owner | Yolanda Sanz |
| Editor / Company | GSMA |

It is our intention to provide a quality product for your use. If you find any errors or omissions, please contact us with your comments. You may notify us at [prd@gsma.com](mailto:prd@gsma.com).

Your comments or suggestions & questions are always welcome.