|  |  |
| --- | --- |
| 3GPP TS 31.213 V16.1.0 (2020-09) | |
| Technical Specification | |
| 3rd Generation Partnership Project;  Technical Specification Group Core Network and Terminals;  Test specification for (U)SIM;  Application Programming Interface (API) for Java Card™  (Release 16) | |
|  | |
|  |  |
|  | |
| The present document has been developed within the 3rd Generation Partnership Project (3GPP TM) and may be further elaborated for the purposes of 3GPP. The present document has not been subject to any approval process by the 3GPPOrganizational Partners and shall not be implemented. This Specification is provided for future development work within 3GPPonly. The Organizational Partners accept no liability for any use of this Specification. Specifications and Reports for implementation of the 3GPP TM system should be obtained via the 3GPP Organizational Partners' Publications Offices. | |

|  |
| --- |
|  |
| ***3GPP***  Postal address  3GPP support office address  650 Route des Lucioles - Sophia Antipolis  Valbonne - FRANCE  Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16  Internet  http://www.3gpp.org |
| ***Copyright Notification***  No part may be reproduced except as authorized by written permission. The copyright and the foregoing restriction extend to reproduction in all media.  © 2020, 3GPP Organizational Partners (ARIB, ATIS, CCSA, ETSI, TSDSI, TTA, TTC).  All rights reserved.  UMTS™ is a Trade Mark of ETSI registered for the benefit of its members  3GPP™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners LTE™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners  GSM® and the GSM logo are registered and owned by the GSM Association |

Contents

Foreword 6

1 Scope 7

2 References 7

3 Definitions, symbols and abbreviations 8

3.1 Definitions 8

3.2 Abbreviations 8

4 Test environment 9

4.1 Applicability 9

4.2 Test environment description 9

4.3 Tests format 10

4.3.1 Test area reference 10

4.3.1.1 Conformance requirements 11

4.3.1.2 Test area files 11

4.3.1.3 Test procedure 11

4.3.1.4 Test coverage 12

4.4 Initial conditions 12

4.5 Package name 12

4.6 AID coding 12

4.7 Test equipment 13

4.7.1 Test tool 13

4.7.2 Interfaces and classes use 13

4.7.3 Util package 14

4.7.4 Java Software Development kit version 14

5 Test plan 14

5.1 Package uicc.usim.access package 14

5.1.1 Interface SIMConstants 14

5.1.2 Interface USIMConstants 14

5.2 Package uicc.usim.toolkit package 14

5.2.1 Interface ToolkitConstants 14

5.2.2 Interface USATEnvelopeHandler 14

5.2.2.1 Method getSecuredDataLength 14

5.2.2.2 Method getSecuredDataOffset 19

5.2.2.3 Method getShortMessageLength 22

5.2.2.4 Method getShortMessageOffset 25

5.2.2.5 Method getTPUDLOffset 29

5.2.2.6 Method getUserDataLength 31

5.2.2.7 Method getItemIdentifier 34

5.2.2.8 Method getChannelIdentifier 35

5.2.2.9 Method getChannelStatus 38

5.2.2.10 Method getSize 40

5.2.2.11 Method getTag 41

5.2.2.12 Method compareValue 42

5.2.2.13 Method copy 44

5.2.2.14 Method copyValue 46

5.2.2.15 Method findAndCompareValue(byte tag, byte[] compareBuffer, short compareOffset) 49

5.2.2.16 Method findAndCompareValue(byte tag, byte occurrence, short valueOffset, byte[] compareBuffer, short compareOffset, short compareLength) 51

5.2.2.17 Method findAndCopyValue(byte tag, byte[] dstBuffer, short dstOffset) 54

5.2.2.18 Method findAndCopyValue(byte tag, byte occurrence, short valueOffset, byte[] dstBuffer, short dstOffset, short dstLength) 56

5.2.2.19 Method findTLV 59

5.2.2.20 Method getCapacity 61

5.2.2.21 Method getLength 62

5.2.2.22 Method getValueByte 63

5.2.2.23 Method getValueLength 64

5.2.2.24 Method getValueShort 65

5.2.3 Interface USATTerminalProfile 66

5.2.4 Class USATEnvelopeHandlerSystem 66

5.2.4.1 Method getTheHandler 66

5.2.5 Interface ToolkitRegistry 67

5.2.5.1 Method clearEvent 67

5.2.5.2 Method isEventSet 69

5.2.5.3 Method setEvent 70

5.2.5.4 Method setEventList 72

5.3 (U)SAT Framework 76

5.3.1 Minimum handler availability 76

5.3.1.1 ProactiveHandler 76

5.3.1.2 ProactiveResponseHandler 81

5.3.1.3 EnvelopeHandler 91

5.3.1.4 EnvelopeResponseHandler 94

5.3.1.5 USATEnvelopeHandler 103

5.3.1.6 Applet triggering with ongoing proactive session 106

5.3.2 Handler integrity 111

5.3.2.1 ProactiveResponseHandler 111

5.3.2.2 EnvelopeHandler 112

5.3.2.3 USATEnvelopeHandler 123

5.3.3 Exception handling 140

5.3.3.1 General Behaviour 140

5.3.3.2 Interaction with Multiple Triggering 142

5.3.4 Applet triggering 143

5.3.4.1 EVENT\_FORMATTED\_SMS\_PP\_ENV 143

5.3.4.2 EVENT\_UNFORMATTED\_SMS\_PP\_ENV 145

5.3.4.3 EVENT\_FORMATTED\_SMS\_PP\_UPD 147

5.3.4.4 EVENT\_UNFORMATTED\_SMS\_PP\_UPD 149

5.3.4.5 EVENT\_FORMATTED\_SMS\_CB 152

5.3.4.6 EVENT\_UNFORMATTED\_SMS\_CB 153

5.3.4.7 EVENT\_FORMATTED\_USSD 154

5.3.4.8 EVENT\_UNFORMATTED\_USSD 156

5.3.5 Envelope response posting 157

5.3.5.1 EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM 157

5.3.6 Toolkit installation 159

5.3.6.1 Minimum security level 159

5.3.6.2 TAR 161

5.3.6.3 Access domain 164

5.3.7 Other parts transferred to (U)SAT framework from API 165

5.3.7.1 A handler is a temporary JCRE Entry Point object 165

5.3.8 Framework security management 166

5.3.8.1 Input data 167

5.3.8.2 Output data 174

5.3.9 Concatenated SMS 175

5.3.9.1 Concatenation processing 175

5.3.9.2 Test area files 176

5.3.9.3 Test coverage 176

5.3.9.4 Test procedure 176

5.3.10 Cell Broadcast Service 178

5.3.10.1 Multiple message reassembling 178

5.3.10.2 Test area files 178

5.3.10.3 Test coverage 178

5.3.11 Concatenated USSD 179

5.3.11.1 Concatenation processing 179

5.3.11.2 Test area files 180

5.3.11.3 Test coverage 180

5.3.11.4 Test procedure 180

Annex A (normative): Class, methods and USATFramework tests acronyms 182

A.1 Toolkit part 182

A.1.1 USATEnvelopeHandler interface 182

A.1.2 USATEnvelopeHandlerSystem method 182

A.1. 3 ToolkitRegistry methods 183

A.2 Acronyms for USATFramework tests 183

A.2.1 Minimum handler availability 183

A.2.2 Handler integrity 183

A.2.3 Applet triggering 183

A.2.4 Exception handling 183

A.2.5 Envelope response posting 184

A.2.6 Toolkit installation 184

A.2.7 Other parts transferred from API to CAT RE 184

A.2.8 Framework security 184

A.2.9 Concatenated SMS 184

A.2.10 Cell Broadcast Service 184

Annex B (normative): Global prepersonalization 185

Annex C (normative): Test file description 186

Annex D (normative): uicc.usim.test.util package, (U)SIM interfaces and testing script example 187

Annex E (normative): Test Area files 188

Annex F (informative): Change history 189

# Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

In the present document, modal verbs have the following meanings:

**shall** indicates a mandatory requirement to do something

**shall not** indicates an interdiction (prohibition) to do something

The constructions "shall" and "shall not" are confined to the context of normative provisions, and do not appear in Technical Reports.

The constructions "must" and "must not" are not used as substitutes for "shall" and "shall not". Their use is avoided insofar as possible, and they are not used in a normative context except in a direct citation from an external, referenced, non-3GPP document, or so as to maintain continuity of style when extending or modifying the provisions of such a referenced document.

**should** indicates a recommendation to do something

**should not** indicates a recommendation not to do something

**may** indicates permission to do something

**need not** indicates permission not to do something

The construction "may not" is ambiguous and is not used in normative elements. The unambiguous constructions "might not" or "shall not" are used instead, depending upon the meaning intended.

**can** indicates that something is possible

**cannot** indicates that something is impossible

The constructions "can" and "cannot" are not substitutes for "may" and "need not".

**will** indicates that something is certain or expected to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

**will not** indicates that something is certain or expected not to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

**might** indicates a likelihood that something will happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

**might not** indicates a likelihood that something will not happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

In addition:

**is** (or any other verb in the indicative mood) indicates a statement of fact

**is not** (or any other negative verb in the indicative mood) indicates a statement of fact

The constructions "is" and "is not" do not indicate requirements.

# 1 Scope

The present document covers the minimum characteristics considered necessary in order to provide compliance to TS 31.130 [2].

The present document describes the technical characteristics and methods of test for testing the (U)SIM API for Java CardTM (TS 31.130 [2]) implemented in the (U)SIM. It specifies the following parts:

- test applicability;

- test environment description;

- tests format;

- test area reference;

- conformance requirements;

- test suite files;

- test procedure;

- test coverage; and

- a description of the associated testing tools that shall be used.

# 2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non‑specific.

- For a specific reference, subsequent revisions do not apply.

- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] ETSI TS 101 220: "Integrated Circuit Cards (ICC); ETSI numbering system for telecommunication; Application providers (AID)".

[2] 3GPP TS 31 130 Release 6: "(U)SIM API for Java Card™"

[3] Void.

[4] 3GPP TS 31.102: "Characteristics of the USIM Application".

[5] 3GPP TS 51.011 Release 4: "Specification of the Subscriber Identity Module- Mobile Equipment (SIM – ME) interface".

[6] 3GPP TS 23.041: "Technical realization of Cell Broadcast Service (CBS)".

[7] Void.

[8] 3GPP TS 31.111: "USIM Application Toolkit (USAT)".

[9] Void.

[10] 3GPP TS 31.115: "Secured packet structure for the (U)SIM Toolkit applications".

[11] 3GPP TS 23.040: "Technical realization of the Short Message Service (SMS)".

[12] Sun Microsystems Java Card™ Specification: "Java Card™ 2.2.1 Application Programming Interface".

[13] Sun Microsystems Java Card™ Specification: "Java Card™ 2.2.1 Runtime Environment (JCRE) Specification".

[14] Sun Microsystems Java Card™ Specification: "Java Card™ 2.2.1 Virtual Machine Specification".

SUN Java Card™ Specifications can be downloaded at <http://java.sun.com/products/javacard>

[15] ETSI TS 102 268 V6.1.0: "Test specification for UICC Application Programming Interface for Java Card (TM) ".

[16] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

# 3 Definitions, symbols and abbreviations

## 3.1 Definitions

For the purposes of the present document, the terms and definitions given in TR 21.905 [16] and the following apply. A term defined in the present document takes precedence over the definition of the same term, if any, in TR 21.905 [16].

**applet installation parameters:** values for applet installation parameters

**Conformance Requirement Reference (CRR):** description of the expected card behaviour according to TS 31.130 [2]

**expected state:** state in which the (U)SIM is supposed to be after the execution of the test procedure applied on the relevant initial conditions

**security parameters:** minimum security requirements defined for the applet installation process

**test area:** set of Test Cases applicable to a specific part (class method, CAT RE behaviour, etc) of the TS 31.130 [2].

**test case:** elementary test that checks for compliance with one or more Conformance Requirement References

**test procedure:** sequence of actions/commands to perform all the test cases defined in a test area

**test source file:** java file containing methods that will load and install test applet in the card, execute and verify the test results, and restore the Default Initial Conditions on the (U)SIM (when possible).

**test toolkit applet:** applet designed to test a specific functionality of the USIM API (TS 31.130 [2])

## 3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TR 21.905 [16] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in TR 21.905 [16].

AID Application IDentifier

APDU Application Protocol Data Unit

API Application Programming Interface

CAT RE Card Application Toolkit Runtime Environment

CRR Conformance requirements Reference

CRRC Conformance requirement Reference Context Error

CRRN Conformance requirement Reference Normal

CRRP Conformance requirement Reference Parameter Error

FFS For Further Study

# 4 Test environment

This clause specifies requirements that shall be met and the testing rules that shall be followed during the test procedure.

## 4.1 Applicability

The tests defined in the present document shall be performed taking into account the services supported by the card as specified in the EFSST file.

The test defined in the present document are applicable to cards implementing TS 31.130 [2] unless otherwise stated.

The tests defined in the present document require that the card support the concatenation process with 2 concatenated SMS.

## 4.2 Test environment description

The general architecture for the test environment is:



## 4.3 Tests format

### 4.3.1 Test area reference

Each test area is referenced as follows:

For API testing:

API Testing: 'API\_[package name]\_[classname]\_[methodname]' where

package name:

uicc.usim.access package: '1'

uicc.usim.toolkit package: '2'

class name:

yyy: 3 letters for each class.

See Annex A for full classes acronyms list.

method name:

zzzz[input parameters]:

See Annex A for full methods name acronyms list.

For Framework testing:

FWK: framework testing

Chapter name:

xxx: 3 letters for each chapter

See annex F for full chapter acronyms list

Subchapter name

yyyy: : 4 letters for each subchapter

See annex F for full subchapter acronyms list

#### 4.3.1.1 Conformance requirements

The conformance requirements are expressed in the following way:

- Method prototype as listed in TS 31.130 [2].

- Normal execution:

- Contains normal execution and correct parameters limit values, each referenced as a Conformance Requirement Reference Normal (CRRN).

- Parameters error:

- Contains parameter errors and incorrect parameter limit values, each referenced as a Conformance Requirement Reference Parameter Error (CRRP).

- Context error:

- Contains errors due to the context the method is used in, each referenced as a Conformance Requirement Reference Context Error (CRRC).

#### 4.3.1.2 Test area files

The files included in the Test Area use the following naming convention:

- Test Source: Test\_[Test Area Reference].java

- Test Applet: [Test Area Reference]\_[Test applet number].java

- Cap File: [Test Area Reference].cap

The applet numbers start from '1'.

The test source shall use common interfaces defined in Annex D.

The Cap File format is described in Java Card™ Virtual Machine Specification [4].

Test files can be run in any order.

All files from the same test area are located in the same subfolder.

#### 4.3.1.3 Test procedure

Each test procedure contains a table to indicate the expected responses form the API and/or the APDU level as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | | | |
| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
|  | *Test Case detailed description* | *API and/or (U)SAT Framework expected behaviour.* | *Expected response at APDU level.* |

#### 4.3.1.4 Test coverage

The table above each test procedure indicates the correspondence between the Conformance Requirements Reference (CRR) and the different test cases.

## 4.4 Initial conditions

The Initial Conditions are a set of general prerequisites for the (U)SIM prior to the execution of testing. For each test procedure described in the present document, the following rules apply to the Initial Conditions:

- unless otherwise stated, the file system and the files' content shall fulfil the requirements described in annex B;

- unless otherwise stated, before installing the applet(s) relevant to the current test procedure, all packages specific to other test procedures shall not be present.

When both statements apply, a test procedure is said to be in the "Default Initial Conditions" state.

## 4.5 Package name

Java packages integrating this Test Suite shall follow this naming convention:

**uicc.usim.test.access.[Test Area Reference]:** Java Card packages containing Test Area References for the TS 31.130 [2] uicc.access package.

**uicc.usim.test.toolkit.[Test Area Reference]:** Java Card packages containing Test Area References for the TS 31.130 [2] uicc.toolkit package.

**uicc.usim.test.usatframework.[Test Area Reference]:** Java Card packages containing Test Area References for the TS 31.130 [2] USAT Framework.

**uicc.usim.test.util:** for the Test util package defined in this Test Suite.

where the Test Area Reference is written in lower case.

EXAMPLE: The package *../uicc.usim.test.access.[Test Area Reference]* creates the following directory structure *../uiccusim//test/access/[Test Area Reference]/Api\_1\_...\_[1..n].\*,*  where '*Api\_1\_...\_[1..n].\**' are the different test applets Java source files used in *[Test Area Reference]*.

## 4.6 AID coding

The AID coding for the Test Packages, Applet classes and Applets shall be as specified in ETSI TS 101 220 [5]. In addition, the following TAR and Application Provider specific data values are defined for use within the present document:

AID coding

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 1 | |  | | Byte 12 | | Byte 13 | | Byte 14 | | Byte 15 | | Byte 16 | |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Application Provider specific data | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | TAR | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Specified in ETSI TS 101 220 [5] | | | | | |

TAR coding (3 bytes/ 24 bits):

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| b1 | |  | | b4 | | b5 | |  | | b8 | | b9 | |  | | b12 | | b13 | |  | | b16 | | b17 | |  | | b24 | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Applet instance number |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Applet class number |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Package number |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | RFU |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Test Part Identifier |

Applet instance number, Applet Class number, Package number:

For package AID, package number shall start from 0 and class and instance numbers shall be 0.

For class AID, package number is the number of the class package, class number shall start from 1 and instance shall be 0.

For instance AID, package and class number are the number of class and package of which instance belongs, and instance number shall start from 1.

Test part Identifier (bits b1-b4):

- 0000 reserved (as TAR= '00.00.00' is reserved for Issuer Security Domain)

- 0010 API uicc.usim.toolkit

- 0101 USAT Framework

- 1110 USIM ADF

- 1111 uicc.util

- other values are RFU

Application Provider specific data (1 byte):

- '00' for Package

- '01' for Applet class

- '02' for Applet Instance

EXAMPLE: The AID of Package uicc.usim.util is 'A0 00 00 00 87 10 05 FF FF FF FF 89 F0 00 00 00'.

## 4.7 Test equipment

These clauses recommend a minimum specification for each of the items of test equipment referenced in the tests.

### 4.7.1 Test tool

This test tool shall meet the following requirements:

- be able to send and receive APDU command to the USIM;

- the result of I/O commands must be presented at the application layer;

- be able to provide results of the tests;

- shall send and/or compare all data specified in test file.

### 4.7.2 Interfaces and classes use

The USIM test tool extends the UICC test tool defined in ETSI TS 102 268 [15]. Then The USIM test tool cannot be run without having implemented the UICC test tool.

The USIM test tool shall use some interfaces and classes, defined in Annex D. They define the only allowed methods to write the test sources.

Interfaces and classes are defined as follow:

- USimToolkitService defines methods to manage toolkit commands,

- USimAPITestService defines methods to send envelopes defined in TS 31.111 [8],

- USimAPITestCardService defines the static method to get a reference of the class implementing all interfaces.

### 4.7.3 Util package

Annex D includes java source code of TestToolkitApplet abstract class of the uicc.usim.util package. Each test applet shall extend this abstract class in order to retrieve test results when selecting it.

### 4.7.4 Java Software Development kit version

Java software development kit (SDK) version supported by Java Card 2.2.1 specifications ([12], [13], [14]) is 1.4.1.

# 5 Test plan

The test plan is divided according to the (U)SIM API specification, that way the tests will follow the class hierarchy for the uicc.usim.toolkit and uicc.usim.access package; for the SIM Toolkit framework this test plan describes the different points that will be tested with the present test specification.

## 5.1 Package uicc.usim.access package

### 5.1.1 Interface SIMConstants

The constants in Java are resolved at compilation time, therefore a runtime test is not useful. No test of constants will be performed.

### 5.1.2 Interface USIMConstants

The constants in Java are resolved at compilation time, therefore a runtime test is not useful. No test of constants will be performed.

## 5.2 Package uicc.usim.toolkit package

### 5.2.1 Interface ToolkitConstants

The constants in Java are resolved at compilation time, therefore a runtime test is not useful. No test of constants will be performed.

### 5.2.2 Interface USATEnvelopeHandler

#### 5.2.2.1 Method getSecuredDataLength

Test Area Reference: Api\_2\_Ueh\_Gsdl

5.2.2.1.1 Conformance requirement

The method with following header shall be compliant to its definition in the API.

public short getSecuredDataLength()

throws uicc.toolkit.ToolkitException

5.2.2.1.1.1 Normal execution

- CRRN1: The method shall return the length of the Secured Data from the Command Packet in the SMS TPDU (simple or concatenated) or Cell Broadcast Page Comprehension TLV contained in the Envelope handler.

- CRRN2: The length is from the first SMS TPDU TLV, USSD String TLV or Cell Broadcast Page Comprehension TLV.

- CRRN3: The length should not include padding bytes.

- CRRN4: The method can be used if the event is EVENT\_FORMATTED\_SMS\_PP\_ENV and if the SMS TP UD is formatted according to TS 31.115 [10] Single or Concatenated Short Message.

- CRRN5: The method can be used if the event is EVENT\_FORMATTED\_SMS\_PP\_UPD and if the SMS TP UD is formatted according to TS 31.115 [10] Single or Concatenated Short Message.

- CRRN6: The method can be used if the event is EVENT\_FORMATTED\_SMS\_CB and if the Cell Broadcast Page is formatted according to TS 31.115 [10].

- CRRN7: The method can be used if the event is EVENT\_FORMATTED\_USSD and if the USSD String id is formatted according to TS31.115 [10]

- CRRN8: If the method is successful and if the event is EVENT\_FORMATTED\_SMS\_PP\_ENV, the selected TLV should be the SMS TPDU TLV.

- CRRN9: If the method is successful and if the event is EVENT\_FORMATTED\_SMS\_PP\_UPD, the selected TLV should be the SMS TPDU TLV.

- CRRN10: If the method is successful and if the event is EVENT\_FORMATTED\_SMS\_CB, the selected TLV should be the Cell Broadcast Page TLV.

- CRRN11: If the method is successful and if the event is EVENT\_FORMATTED\_USSD, the selected TLV should be the USSD String TLV.

5.2.2.1.1.2 Parameter errors

No requirements.

5.2.2.1.1.3 Context errors

- CRRC1: The method shall throw ToolkitException.UNAVAILABLE\_ELEMENT in case of unavailable SMS TPDU TLV, USSD String TLV element or Cell Broadcast Page Comprehension TLV.

- CRRC2: The method shall throw ToolkitException.UNAVAILABLE\_ELEMENT in case of wrong data format.

5.2.2.1.2 Test area files

Specific triggering:

- FORMATTED SMS CB.

- UNFORMATTED SMS CB.

- FORMATTED SMS PP ENV.

- UNFORMATED SMS PP ENV.

- FORMATTED SMS PP UPD.

- FORMATTED USSD ENV.

- UNFORMATTED USSD ENV.

- For Formatted triggering if CC/RC/DS is used, the security parameters are the one defined in clause "5.3.8 Framework Security Management".

Test Source: Test\_Api\_2\_Ueh\_Gsdl.java

Test Applet: Api\_2\_Ueh\_Gsdl\_1.java

Cap File: Api\_2\_Ueh\_Gsdl.cap

5.2.2.1.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | 1 to 42 |
| N2 | 13, 30 |
| N3 | 6, 7, 23, 24, 37, 38 |
| N4 | 1 to 17 |
| N5 | 18 to 34 |
| N6 | 35 to 42 |
| N7 | 43 to 56 |
| N8 | 17 |
| N9 | 34 |
| N10 | 42 |
| N11 | 56 |
| C1 | 57 |
| C2 | 58 |
| C3 | 59 |

5.2.2.1.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
|  | **FORMATTED SMS PP ENV Triggering** |  |  |
| 1 | Test with FORMATTED\_SMS\_PP\_ENV and TP-OA length of 2 | Returns 0x002A |  |
| 2 | Test with TP-OA length of 6 | Returns 0x002A |  |
| 3 | Test with TP-OA length of 12 | Returns 0x002A |  |
| 4 | Test with RC/CC/DS length of 0 | Returns 0x0010 |  |
| 5 | Test with RC/CC/DS length of 8 | Returns 0x0010 |  |
| 6 | Test with PCNTR = 0 | Returns 0x0010 |  |
| 7 | Test with PCNTR = 7 (ciphering shall be used) | Returns 0x0003 |  |
| 8 | Test with Secured Data Length = 00 | Returns 0x0000 |  |
| 9 | Test with Secured Data Length = 0x33 | Returns 0x0033 |  |
| 10 | Test with Secured Data Length = 0x6C (UDL = 0x7F) | Returns 0x006C |  |
| 11 | Test with Secured Data Length = 0x6D (UDL = 0x80) | Returns 0x006D |  |
| 12 | Test with Secured Data Length = maximum length for one envelope : 0x79 (UDL = 0x8C) | Returns 0x0079 |  |
| 13 | Verify it is the first TPDU TLV:  Send a SMS PP with 2 TPDU TLV and inside two different secured data lengths: 5 and 10 | Returns 0x0005 |  |
| 14 | Test with secured data length = 0x7F (2 concatenated envelopes are needed) | Returns 0x007F |  |
| 15 | Test with secured data length = 0x80 (2 concatenated envelopes are needed) | Returns 0x0080 |  |
| 16 | Test with secured data length = maximum length for 2 concatenated envelopes : 0xFA | Returns 0x00FA |  |
| 17 | Test with FORMATTED\_SMS\_PP\_ENV  Verify after call of the method the current TLV is the TPDU TLV:  findTLV device identities, getSecuredDataLength and then getValueByte to verify that the current TLV is the TPDU TLV | getValueByte returns 0x0040 |  |
|  | **FORMATTED SMS PP UPD Triggering** |  |  |
| 18 | Same test as 1 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x002A |  |
| 19 | Same test as 2 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x002A |  |
| 20 | Same test as 3 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x002A |  |
| 21 | Same test as 4 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x0010 |  |
| 22 | Same test as 5 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x0010 |  |
| 23 | Same test as 6 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x0010 |  |
| 24 | Same test as 7 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x0003 |  |
| 25 | Same test as 8 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x0000 |  |
| 26 | Same test as 9 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x0033 |  |
| 27 | Same test as 10 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x006C |  |
| 28 | Same test as 11 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x006D |  |
| 29 | Same test as 12 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x0079 |  |
| 30 | Same test as 13 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x0005 |  |
| 31 | Test with secured data length = 0x7F (2 concatenated envelopes are needed) | Returns 0x007F |  |
| 32 | Test with secured data length = 0x80 (2 concatenated envelopes are needed) | Returns 0x0080 |  |
| 33 | Test with secured data length = maximum length for 2 concatenated envelopes : 0xFA | Returns 0x00FA |  |
| 34 | Test with FORMATTED\_SMS\_PP\_UPD  Verify after call of the method the current TLV is the TPDU TLV:  findTLV device identities, getSecuredDataLength and then getValueByte to verify that the current TLV is the TPDU TLV | getValueByte returns 0x0040 |  |
|  | **FORMATTED SMS CB Triggering** |  |  |
| 35 | Same test as 4 but with FORMATTED\_SMS\_CB | Returns 0x0010 |  |
| 36 | Same test as 5 but with FORMATTED\_SMS\_CB | Returns 0x0010 |  |
| 37 | Same test as 6 but with FORMATTED\_SMS\_CB | Returns 0x0010 |  |
| 38 | Same test as 7 but with FORMATTED\_SMS\_CB | Returns 0x0003 |  |
| 39 | Same test as 8 but with FORMATTED\_SMS\_CB | Returns 0x0000 |  |
| 40 | Same test as 9 but with FORMATTED\_SMS\_CB | Returns 0x0033 |  |
| 41 | Same test as 12 but with maximum secured data length: 0x42, and FORMATTED\_SMS\_CB | Returns 0x0042 |  |
| 42 | Test with FORMATTED\_SMS\_CB  Verify after call of the method the current TLV is the Cell Broadcast Page TLV:  findTLV device identities, getSecuredDataLength and then getValueByte to verify that the current TLV is the Cell Broadcast Page TLV | getValueByte returns 0x00 |  |
|  | **FORMATTED USSD Triggering** |  |  |
| 43 | Test with formatted USSD and RC/CC/DS length of 0 | Returns 0x0010 |  |
| 44 | Test with RC/CC/DS length of 8 | Returns 0x0010 |  |
| 45 | Test with PCNTR = 0 | Returns 0x0010 |  |
| 46 | Test with PCNTR = 7 (ciphering shall be used) | Returns 0x0003 |  |
| 47 | Test with Secured Data Length = 00 | Returns 0x0000 |  |
| 48 | Test with Secured Data Length = 0x33 | Returns 0x0033 |  |
| 49 | Test with Secured Data Length = 0x6C (UDL = 0x7F) | Returns 0x006C |  |
| 50 | Test with Secured Data Length = 0x6D (UDL = 0x80) | Returns 0x006D |  |
| 51 | Test with Secured Data Length = maximum length for one envelope : 0x79 (UDL = 0x8C) | Returns 0x0079 |  |
| 52 | Verify it is the first String TLV:  Send a USSD with 2 USSD String TLV and inside two different secured data lengths: 5 and 10 | Returns 0x0005 |  |
| 53 | Test with secured data length = 0x7F (2 concatenated envelopes are needed) | Returns 0x007F |  |
| 54 | Test with secured data length = 0x80 (2 concatenated envelopes are needed) | Returns 0x0080 |  |
| 55 | Test with secured data length = maximum length for 2 concatenated envelopes : 0xFA | Returns 0x00FA |  |
| 56 | Test with FORMATTED\_USSD  Verify after call of the method the current TLV is the USSD String TLV:  findTLV device identities, getSecuredDataLength and then getValueByte to verify that the current TLV is the USSD String TLV | getValueByte returns 0x0040 |  |
|  | **Error tests** |  |  |
| 57 | Send an envelope SMS CB, getSecuredDataLength | ToolkitException.UNAVAILABLE\_ELEMENT |  |
| 58 | Send an envelope SMS PP unformatted | ToolkitException.UNAVAILABLE\_ELEMENT |  |
| 59 | Send an envelope USSD unformatted | ToolkitException.UNAVAILABLE\_ELEMENT |  |

#### 5.2.2.2 Method getSecuredDataOffset

Test Area Reference: Api\_2\_Ueh\_Gsdo

5.2.2.2.1 Conformance requirement

The method with following header shall be compliant to its definition in the API.

public short getSecuredDataOffset()

throws uicc.toolkit.ToolkitException

5.2.2.2.1.1 Normal execution

- CRRN1: The method shall return the offset of the secured data first byte contained in a SMS TPDU TLV or USSD String TLV.

- CRRN2: The offset is from the first SMS TPDU TLV or USSD String TLV.

- CRRN3: The method can be used if the event is EVENT\_FORMATTED\_SMS\_PP\_ENV and if the SMS TP-UD is formatted according to TS 31.115 [10].

- CRRN4: The method can be used if the event is EVENT\_FORMATTED\_SMS\_PP\_UPD and if the SMS TP-UD is formatted according to TS 31.115 [10].

- CRRN5: The method can be used if the event is EVENT\_FORMATTED\_SMS\_CB and if the Cell Broadcast Page is formatted according to TS 31.115 [10].

- CRRN6: If the method is successful and if the event is EVENT\_FORMATTED\_SMS\_PP\_ENV, the selected TLV should be the SMS TPDU TLV.

- CRRN7: If the method is successful and if the event is EVENT\_FORMATTED\_SMS\_PP\_UPD, the selected TLV should be the SMS TPDU TLV.

- CRRN8: If the method is successful and if the event is EVENT\_FORMATTED\_SMS\_CB, the selected TLV should be the Cell Broadcast Page TLV.

- CRRN9: If the method is successful and if the event is EVENT\_FORMATTED\_USSD, the selected TLV should be the USSD String TLV.

- CRRN10: If the Secured Data length is zero the value returned shall be the offset of the first byte following the TS 31.115 [10] Command Packet structure.

5.2.2.2.1.2 Parameter errors

No requirements.

5.2.2.2.1.3 Context errors

- CRRC1: The method shall throw ToolkitException.UNAVAILABLE\_ELEMENT in case of unavailable SMS TPDU TLV or USSD String TLV element.

- CRRC2: The method shall throw ToolkitException.UNAVAILABLE\_ELEMENT in case of wrong data format.

5.2.2.2.2 Test area files

Specific triggering:

- FORMATTED SMS CB.

- UNFORMATTED SMS CB.

- FORMATTED SMS PP UPD.

- FORMATTED SMS PP ENV.

- UNFORMATED SMS PP ENV.

- FORMATTED USSD ENV.

- UNFORMATTED USSD ENV.

- For Formatted triggering if CC/RC/DS is used, the security parameters are the one defined in clause "5.3.8 Framework Security Management".

Test Source: Test\_Api\_2\_Ueh\_Gsdo.java

Test Applet: Api\_2\_Ueh\_Gsdo\_1.java

Cap File: Api\_2\_Ueh\_Gsdo.cap

5.2.2.2.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | 1 to 20 |
| N2 | 5, 13 |
| N3 | 1 to 8 |
| N4 | 9 to 16 |
| N5 | 17, 18, 19, 20 |
| N6 | 7 |
| N7 | 15 |
| N8 | 20 |
| N9 | 21 to 25 |
| N10 | 6, 14, 19 |
| C1 | 26 |
| C2 | 27 |

5.2.2.2.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
|  | **FORMATTED SMS PP ENV triggering** |  |  |
| 1 | Test with TP-OA length of 2 and RC/CC/DS length is 0 | Returns 0x21 |  |
| 2 | Test with TP-OA length of 6 and RC/CC/DS length is 0 | Returns 0x23 |  |
| 3 | Test with TP-OA length of 12 and RC/CC/DS length is 0 | Returns 0x26 |  |
| 4 | Test with RC/CC/DS length of 8 and TP-OA length is 2 | Returns 0x29 |  |
| 5 | Send a SMS PP with 2 TPDU TLV and inside two different secured data offsets | Returns 0x24 ( the first offset ) |  |
| 6 | Same test as 1 but without any secured data | Returns 0x21 |  |
| 7 | Test with FORMATTED\_SMS\_PP ENV  Verify after call of the method the current TLV is the TPDU TLV:  findTLV device identities, getSecuredDataOffset and then getValueByte to verify that the current TLV is the TPDU TLV | Returns 0x40 |  |
| 8 | Same test as 1, but with a concatenated SMS (2 Short Messages and maximum Secured Data Length = 0x00FA) | Returns 0x21 |  |
|  | **FORMATTED SMS PP UPR triggering** |  |  |
| 9 | Same test as 1 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x21 |  |
| 10 | Same test as 2 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x23 |  |
| 11 | Same test as 3 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x26 |  |
| 12 | Same test as 4 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x29 |  |
| 13 | Same test as 5 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x24 ( the first offset ) |  |
| 14 | Same test as 6 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x21 |  |
| 15 | Test with FORMATTED\_SMS\_PP UPD  Verify after call of the method the current TLV is the TPDU TLV:  findTLV device identities, getSecuredDataOffset and then getValueByte to verify that the current TLV is the TPDU TLV | Returns 0x40 |  |
| 16 | Same test as 8, but with a concatenated SMS (2 Short Messages and maximum Secured Data Length = 0x00FA) | Returns 0x21 |  |
|  | **FORMATTED SMS CB triggering** |  |  |
| 17 | Same test as 2 but with FORMATTED\_SMS\_CB | Returns 0x16 |  |
| 18 | Same test as 4 but with FORMATTED\_SMS\_CB | Returns 0x1E |  |
| 19 | Same test as 6 but with FORMATTED\_SMS\_CB | Returns 0x16 |  |
| 20 | Test with FORMATTED\_SMS\_CB  Verify after call of the method the current TLV is the Cell Broadcast Page TLV:  findTLV device identities, getSecuredDataOffset and then getValueByte to verify that the current TLV is the Cell Broadcast Page TLV | Returns 0x00 |  |
|  | **FORMATTED USSD triggering** |  |  |
| 21 | Test with RC/CC/DS length of 8 | Returns 0x29 |  |
| 22 | Send a USSD with 2 USSD StringTLV and inside two different secured data offsets | Returns 0x24 ( the first offset ) |  |
| 23 | Same test as 1 but without any secured data | Returns 0x21 |  |
| 24 | Test with FORMATTED\_USSD  Verify after call of the method the current TLV is the String TLV:  findTLV device identities, getSecuredDataOffset and then getValueByte to verify that the current TLV is the StringTLV | Returns 0x40 |  |
| 25 | Same test as 1, but with a concatenated SMS (2 USSD message and maximum Secured Data Length = 0x00FA) | Returns 0x21 |  |
|  | **UNFORMATTED Triggering** |  |  |
| 26 | Send an UNFORMATTED SMS CB envelope, getSecuredDataOffset | ToolkitException.UNAVAILABLE\_ELEMENT |  |
| 27 | Send an UNFORMATTED SMS PP envelope, getSecuredDataOffset | ToolkitException.UNAVAILABLE\_ELEMENT |  |
| 28 | Send an UNFORMATTED USSD envelope, getSecuredDataOffset | ToolkitException.UNAVAILABLE\_ELEMENT |  |

#### 5.2.2.3 Method getShortMessageLength

Test Area Reference: Api\_2\_Ueh\_Gsml

5.2.2.3.1 Conformance requirement

The method with following header shall be compliant to its definition in the API.

public short getShortMessageLength()

throws uicc.toolkit.ToolkitException

5.2.2.3.1.1 Normal execution

- CRRN1: The method shall return the length of the Short Message from the User Data part in the SMS TPDU (simple or concatenated) or Cell Broadcast Page Comprehension TLV contained in the USATEnvelopeHandler.

- CRRN2: The length is from the first SMS TPDU TLV or Cell Broadcast Page Comprehension TLV.

- CRRN3: The length should not include padding bytes.

- CRRN4: The length should not include the UDH if any.

- CRRN5: The method can be used if the event is EVENT\_FORMATTED\_SMS\_PP\_ENV or EVENT\_UNFORMATTED\_SMS\_PP.

- CRRN6: The method can be used if the event is EVENT\_FORMATTED\_SMS\_PP\_UPD or EVENT\_UNFORMATTED\_SMS\_UPD.

- CRRN7: The method can be used if the event is EVENT\_FORMATTED\_SMS\_CB or EVENT\_UNFORMATTED\_SMS\_CB.

- CRRN8: If the method is successful and if the event is EVENT\_FORMATTED\_SMS\_PP\_ENV or EVENT\_UNFORMATTED\_SMS\_PP, the selected TLV should be the SMS TPDU TLV.

- CRRN9: If the method is successful and if the event is EVENT\_FORMATTED\_SMS\_PP\_UPD or EVENT\_UNFORMATTED\_SMS\_UPD, the selected TLV should be the SMS TPDU TLV.

- CRRN10: If the method is successful and if the event is EVENT\_FORMATTED\_SMS\_CB or EVENT\_UNFORMATTED\_SMS\_CB, the selected TLV should be the Cell Broadcast Page TLV.

- CRRN11: If the Short Message Length is zero, no exception shall be thrown.

5.2.2.3.1.2 Parameter errors

No requirements.

5.2.2.3.1.3 Context errors

- CRRC1: The method shall throw ToolkitException.UNAVAILABLE\_ELEMENT in case of unavailable SMS TPDU TLV element or Cell Broadcast Page Comprehension TLV.

- CRRC2: The method shall throw ToolkitException.UNAVAILABLE\_ELEMENT in case of wrong data format.

5.2.2.3.2 Test area files

Specific triggering:

- FORMATTED SMS CB.

- UNFORMATTED SMS CB.

- FORMATTED SMS PP UPD.

- UNFORMATTED SMS PP UPD.

- FORMATTED SMS PP ENV.

- UNFORMATTED SMS PP ENV.

- UNRECOGNIZED\_ENVELOPE.

- For Formatted triggering if CC/RC/DS is used, the security parameters are the one defined in clause "5.3.8 Framework Security Management".

Test Source: Test\_Api\_2\_Ueh\_Gsml.java

Test Applet: Api\_2\_Ueh\_Gsml\_1.java

Cap File: Api\_2\_Ueh\_Gsml.cap

5.2.2.3.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | 1 to 46 |
| N2 | 8, 18, 26, 34, 41, 45 |
| N3 | 4, 14, 22 |
| N4 | 1 to 27 |
| N5 | 1 to 10 and 28 to 35 |
| N6 | 11 to 20 and 36 to 43 |
| N7 | 21 to 27 and 44 to 46 |
| N8 | 9, 35 |
| N9 | 19, 43 |
| N10 | 27, 46 |
| N11 | 6, 16, 24, 30, 38 |
| C1 | 47 |
| C2 | Not applicable |

5.2.2.3.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
|  | FORMATTED SMS PP ENV Triggering |  |  |
| 1 | Test with FORMATTED\_SMS\_PP\_ENV and TP-OA length of 2 and secured data length of 0x2A with no padding byte (PCNTR = 0, no RC/CC/DS) | Returns 0x003A |  |
| 2 | Test with TP-OA length of 12 and secured data length of 0x2A with no padding byte (PCNTR = 0, no RC/CC/DS) | Returns 0x003A |  |
| 3 | Test with PCNTR = 0, no RC/CC/DS and data length of 0x10 | Returns 0x0020 |  |
| 4 | Test with PCNTR = 7, no RC/CC/DS and data length of 0x03 (ciphering shall be used) | Returns 0x001A |  |
| 5 | Test with PCNTR = 0, with RC/CC/DS length of 8 and secured data length of 0x10 | Returns 0x0028 |  |
| 6 | Test with PCNTR = 0, no RC/CC/DS and SecuredDataLength = 00 | Returns 0x0010 |  |
| 7 | Test with PCNTR = 0, no RC/CC/DS and UserDataLength = maximum length (0x8C) for a single SMS | Returns 0x0089 |  |
| 8 | Verify it is the first TPDU TLV:  Send a SMS PP with 2 TPDU TLV with two different user data lengths: 0x18 and 0x23 (with PCNTR = 0, no RC/CC/DS) | Returns 0x0015 |  |
| 9 | Send envelope SMS-PP Formatted.  FindTLV() with TAG\_DEVICE\_IDENTITIES. getShortMessageLength() and then getValueByte() with offset 0 | getValueByte() returns 0x40(TS 23.040 [11] first byte) |  |
| 10 | Test with UserDataLength = maximum length (0x010D) with 2 concatenated SMS | Returns 0x010A |  |
|  | FORMATTED SMS PP UPD Triggering |  |  |
| 11 | Test with FORMATTED\_SMS\_PP\_UPD and TP-OA length of 2 and secured data length of 0x2A with no padding byte (PCNTR = 0) | Returns 0x003A |  |
| 12 | Test with TP-OA length of 12 and secured data length of 0x2A with no padding byte (PCNTR = 0) | Returns 0x003A |  |
| 13 | Test with PCNTR = 0, no RC/CC/DS and data length of 0x10 | Returns 0x0020 |  |
| 14 | Test with PCNTR = 7, no RC/CC/DS and data length of 0x03 (ciphering shall be used) | Returns 0x001A |  |
| 15 | Test with PCNTR = 0, with RC/CC/DS length of 8 and secured data length of 0x10 | Returns 0x0028 |  |
| 16 | Test with PCNTR = 0, no RC/CC/DS and SecuredDataLength = 00 | Returns 0x0010 |  |
| 17 | Test with PCNTR = 0, no RC/CC/DS and UserDataLength = maximum length (0x8C) for a single SMS | Returns 0x0089 |  |
| 18 | Verify it is the first TPDU TLV:  Send a SMS PP with 2 TPDU TLV with two different user data lengths: 0x18 and 0x23 (with PCNTR = 0, no RC/CC/DS) | Returns 0x0015 |  |
| 19 | Send envelope SMS-PP Formatted.  FindTLV() with TAG\_DEVICE\_IDENTITIES. GetShortMessageLength() and then getValueByte() with offset 0 | GetValueByte() returns 0x40(TS 23.040 [11] first byte) |  |
| 20 | Test with UserDataLength = maximum length (0x010D) with 2 concatenated SMS | Returns 0x0010A |  |
|  | FORMATTED SMS CB Triggering |  |  |
| 21 | Test with PCNTR = 0, no RC/CC/DS and data length of 0x10 | Returns 0x0052 |  |
| 22 | Test with PCNTR = 7, no RC/CC/DS and data length of 0x03 (ciphering shall be used) | Returns 0x0052 |  |
| 23 | Test with PCNTR = 0, with RC/CC/DS length of 8 and secured data length of 0x10 | Returns 0x0052 |  |
| 24 | Test with PCNTR = 0, no RC/CC/DS and SecuredDataLength = 00 | Returns 0x0052 |  |
| 25 | Test with PCNTR = 0, no RC/CC/DS and UserDataLength = maximum length (0x58) for a single SMS CB | Returns 0x0052 |  |
| 26 | Verify it is the first Cell Broadcast Page TLV:  Send a SMS CB with 2 Cell Broadcast Page TLV with two different user data lengths: 0x18 and 0x23 (with PCNTR = 0, no RC/CC/DS) | Returns 0x0052 and GetValueShort(6) returns 0x0016 (UDL of first Page TLV) |  |
| 27 | Send envelope SMS-CB Formatted.  FindTLV() with TAG\_DEVICE\_IDENTITIES. GetShortMessageLength() and then getValueByte() with offset 0 | GetValueByte() returns 0x00 (TS 23.041 first byte) |  |
|  | UNFORMATTED SMS PP ENV Triggering |  |  |
| 28 | Test with UNFORMATTED\_SMS\_PP\_ENV and TP-OA length of 2, and user data length of 0x3D | Returns 0x003D |  |
| 29 | Test with TP-OA length of 12, and user data length of 0x3D | Returns 0x003D |  |
| 30 | Test with UserDataLength = 0x00 | Returns 0x0000 with no exception |  |
| 31 | Test with UserDataLength = 0x7F | Returns 0x007F |  |
| 32 | Test with UserDataLength = 0x80 | Returns 0x0080 |  |
| 33 | Test with UserDataLength = maximum length: 0x8C for a single SMS | Returns 0x008C |  |
| 34 | Verify it is the first TPDU TLV:  Send a SMS PP with 2 TPDU TLV with two different user data lengths: 0x18 and 0x23 | Returns 0x0018 |  |
| 35 | Send envelope SMS-PP Unformatted.  FindTLV() with TAG\_DEVICE\_IDENTITIES. getShortMessageLength() and then getValueByte() with offset 0 (first user data = 0x55) | GetValueByte() returns 0x00  (TS 23.040 [11] first byte) |  |
|  | UNFORMATTED SMS PP UPD Triggering |  |  |
| 36 | Test with UNFORMATTED\_SMS\_PP\_UPD and TP-OA length of 2, and user data length of 0x3D | Returns 0x003D |  |
| 37 | Test with TP-OA length of 12, and user data length of 0x3D | Returns 0x003D |  |
| 38 | Test with UserDataLength = 0x00 | Returns 0x0000 with no exception |  |
| 39 | Test with UserDataLength = 0x7F | Returns 0x007F |  |
| 40 | Test with UserDataLength = 0x80 | Returns 0x0080 |  |
| 41 | Verify it is the first TPDU TLV:  Send a SMS PP with 2 TPDU TLV with two different user data lengths: 0x18 and 0x23 | Returns 0x0018 |  |
| 42 | Test with UserDataLength = maximum length: 0x8C for a single SMS | Returns 0x008C |  |
| 43 | Send envelope SMS-PP Formatted.  FindTLV() with TAG\_DEVICE\_IDENTITIES. GetShortMessageLength() and then getValueByte() with offset 0 | GetValueByte() returns 0x00 (TS 23.040 [11] first byte) |  |
|  | UNFORMATTED SMS CB Triggering |  |  |
| 44 | Test with UNFORMATTED\_SMS\_CB | Returns 0x0052 |  |
| 45 | Verify it is the first Cell Broadcast Page TLV:  Send a SMS CB with 2 Cell Broadcast Page TLV with two different user data lengths: 0x58 and 0x23 | Returns 0x0052 |  |
| 46 | Send envelope SMS-CB Formatted.  FindTLV() with TAG\_DEVICE\_IDENTITIES. GetShortMessageLength() and then getValueByte() with offset 0 | GetValueByte() returns 0x00 (TS 23.041 first byte) |  |
|  | Unrecognized Envelope Triggering |  |  |
| 47 | Send an Unrecognized Envelope with neither TPDU TLV nor Cell Broadcast Page TLV, then call the method. | ToolkitException.UNAVAILABLE\_ELEMENT |  |

#### 5.2.2.4 Method getShortMessageOffset

Test Area Reference: Api\_2\_Ueh\_Gsmo

5.2.2.4.1 Conformance requirement

The method with following header shall be compliant to its definition in the API.

public short getShortMessageOffset()

throws uicc.toolkit.ToolkitException

5.2.2.4.1.1 Normal execution

- CRRN1: The method shall return the offset of the Short Message first byte contained in the User Data part of the SMS TPDU TLV contained in the USATEnvelopeHandler.

- CRRN2: The offset is from the first SMS TPDU TLV.

- CRRN3: The method can be used if the event is EVENT\_FORMATTED\_SMS\_PP\_ENV or EVENT\_UNFORMATTED\_SMS\_PP\_ENV.

- CRRN4: The method can be used if the event is EVENT\_FORMATTED\_SMS\_PP\_UPD or EVENT\_UNFORMATTED\_SMS\_PP\_UPD.

- CRRN5: The method can be used if the event is EVENT\_FORMATTED\_SMS\_CB or EVENT\_UNFORMATTED\_SMS\_CB.

- CRRN6: If the method is successful and if the event is EVENT\_FORMATTED\_SMS\_PP\_ENV or EVENT\_UNFORMATTED\_SMS\_PP\_ENV, the selected TLV should be the SMS TPDU TLV.

- CRRN7: If the method is successful and if the event is EVENT\_FORMATTED\_SMS\_PP\_UPD or EVENT\_UNFORMATTED\_SMS\_PP\_UPD, the selected TLV should be the SMS TPDU TLV.

- CRRN8: If the method is successful and if the event is EVENT\_FORMATTED\_SMS\_CB or EVENT\_UNFORMATTED\_SMS\_CB, the selected TLV should be the Cell Broadcast Page TLV.

- CRRN9: The method returns the offset of the first byte after the UDH, if any.

5.2.2.4.1.2 Parameter errors

No requirements.

5.2.2.4.1.3 Context errors

- CRRC1: The method shall throw ToolkitException.UNAVAILABLE\_ELEMENT in case of unavailable SMS TPDU TLV or Cell Broadcast Page Comprehension TLV element.

- CRRC2: The method shall throw ToolkitException.UNAVAILABLE\_ELEMENT in case of wrong data format.

5.2.2.4.2 Test area files

Specific triggering:

- FORMATTED SMS CB.

- UNFORMATTED SMS CB.

- FORMATTED SMS PP UPD.

- UNFORMATTED SMS PP UPD.

- FORMATED SMS PP ENV.

- UNFORMATED SMS PP ENV.

- UNRECOGNIZED\_ENVELOPE.

- For Formatted triggering if CC/RC/DS is used, the security parameters are the one defined in clause "5.3.8 Framework Security Management".

Test Source: Test\_Api\_2\_Ueh\_Gsmo.java

Test Applet: Api\_2\_Ueh\_Gsmo\_1.java

Cap File: Api\_2\_Ueh\_Gsmo.cap

5.2.2.4.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | 1 to 30 |
| N2 | 4, 11, 21, 26 |
| N3 | 1 to 7 and 19 to 23 |
| N4 | 8 to 14 and 24 to 28 |
| N5 | 15 to 18, 29, 30 |
| N6 | 6, 22 |
| N7 | 13, 27 |
| N8 | 18, 30 |
| N9 | 1 to 18, 23, 28 |
| C1 | 31 |
| C2 | Not applicable |

5.2.2.4.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
|  | **FORMATTED SMS PP ENV triggering** |  |  |
| 1 | Test with TP-OA length of 2 and RC/CC/DS length is 0 | Returns 0x11 |  |
| 2 | Test with TP-OA length of 12 and RC/CC/DS length is 0 | Returns 0x16 |  |
| 3 | Test with RC/CC/DS length of 8 and TP-OA length is 2 | Returns 0x11 |  |
| 4 | Send a SMS PP with 2 TPDU TLV and inside two different secured data offsets | Returns 0x14 ( the first offset ) |  |
| 5 | Same test as 1 but without any secured data | Returns 0x11 |  |
| 6 | Test with FORMATTED\_SMS\_PP ENV  Verify after call of the method the current TLV is the TPDU TLV:  findTLV device identities, getSecuredDataOffset and then getValueByte to verify that the current TLV is the TPDU TLV | Returns 0x40 |  |
| 7 | Same test as 1, but with a concatenated SMS (2 Short Messages and maximum Secured Data Length = 0x00FA) | Returns 0x15 |  |
|  | **FORMATTED SMS PP UPD triggering** |  |  |
| 8 | Same test as 1 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x11 |  |
| 9 | Same test as 2 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x16 |  |
| 10 | Same test as 3 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x11 |  |
| 11 | Same test as 4 but with FORMATTED\_SMS\_PP\_UPD | Returns0x14 ( the first offset ) |  |
| 12 | Same test as 5 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x11 |  |
| 13 | Test with FORMATTED\_SMS\_PP UPD  Verify after call of the method the current TLV is the TPDU TLV:  findTLV device identities, getSecuredDataOffset and then getValueByte to verify that the current TLV is the TPDU TLV | Returns 0x40 |  |
| 14 | Same test as 10, but with a concatenated SMS (2 Short Messages and maximum Secured Data Length = 0x00FA) | Returns 0x21 |  |
|  | **FORMATTED SMS CB triggering** |  |  |
| 15 | Same test as 1 but with FORMATTED\_SMS\_CB | Returns 0x06 |  |
| 16 | Same test as 3 but with FORMATTED\_SMS\_CB | Returns 0x06 |  |
| 17 | Same test as 5 but with FORMATTED\_SMS\_CB | Returns 0x06 |  |
| 18 | Test with FORMATTED\_SMS\_CB  Verify after call of the method the current TLV is the Cell Broadcast Page TLV:  FindTLV() device identities, getSecuredDataOffset() and then getValueByte() to verify that the current TLV is the Cell Broadcast Page TLV | Returns 0x00 |  |
|  | **UNFORMATTED SMS PP ENV triggering** |  |  |
| 19 | Test with TP-OA length of 2 | Returns 0x0E |  |
| 20 | Test with TP-OA length of 12 | Returns 0x13 |  |
| 21 | Send a SMS PP with 2 TPDU TLV and inside two different UDL offsets | Returns 0x11 (the first offset) |  |
| 22 | Test with UNFORMATTED\_SMS\_PP ENV  Verify after call of the method the current TLV is the TPDU TLV:  findTLV device identities, getShortMessageOffset and then getValueByte to verify that the current TLV is the TPDU TLV | Returns 00 |  |
| 23 | Same test as 19, but with a concatenated SMS (2 Short Messages and maximum User Data Length = 0x0102) | Returns 0x0E or 0x0F (depending of UDHI implementation) |  |
|  | **UNFORMATTED SMS PP UPD triggering** |  |  |
| 24 | Same test as 19 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x0E |  |
| 25 | Same test as 20 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x13 |  |
| 26 | Same test as 21 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x11 ( the first offset ) |  |
| 27 | Test with FORMATTED\_SMS\_UPD  Verify after call of the method the current TLV is the TPDU TLV:  findTLV device identities, getShortMessageOffset and then getValueByte to verify that the current TLV is the TPDU TLV | Returns 0x11 |  |
| 28 | Same test as 25, but with a concatenated SMS (2 Short Messages and maximum Secured Data Length = 0x00FA) | Returns 0x0E or 0x0F (depending of UDHI implementation) |  |
|  | UNFORMATTED SMS CB Triggering |  |  |
| 29 | Test with UNFORMATTED\_SMS\_CB | Returns 0x06 |  |
| 30 | Send envelope SMS-CB Unformatted.  FindTLV() with TAG\_DEVICE\_IDENTITIES. getShortMessageOffset() and then getValueByte() with offset 0 | GetValueByte() returns 0x00 (TS 23.041 first byte) |  |
|  | Unrecognized Envelope Triggering |  |  |
| 31 | Send an Unrecognized Envelope with neither TPDU TLV nor Cell Broadcast Page TLV, then call the method. | ToolkitException.UNAVAILABLE\_ELEMENT |  |

#### 5.2.2.5 Method getTPUDLOffset

Test Area Reference: Api\_2\_Ueh\_Gtpo

5.2.2.5.1 Conformance requirement

The method with following header shall be compliant to its definition in the API.

public short getTPUDLOffset()

throws uicc.toolkit.ToolkitException

5.2.2.5.1.1 Normal execution

- CRRN1: The method shall return the TPUDL offset in a SMS TPDU TLV.

- CRRN2: The offset is from the first SMS TPDU TLV.

- CRRN3: The method can be used if the event is EVENT\_FORMATTED\_SMS\_PP\_ENV.

- CRRN4: The method can be used if the event is EVENT\_FORMATTED\_SMS\_PP\_UPD.

- CRRN5: The method can be used if the event is EVENT\_UNFORMATTED\_SMS\_PP\_ENV.

- CRRN6: The method can be used if the event is EVENT\_UNFORMATTED\_SMS\_PP\_UPD.

- CRRN7: If the method is successful, the selected TLV should be the SMS TPDU TLV.

5.2.2.5.1.2 Parameter errors

No requirements.

5.2.2.5.1.3 Context errors

- CRRC1: The method shall throw ToolkitException.UNAVAILABLE\_ELEMENT in case of unavailable SMS TPDU TLV element.

- CRRC2: The method shall throw ToolkitException.UNAVAILABLE\_ELEMENT if the TPUDL field does not exist.

5.2.2.5.2 Test area files

Specific triggering:

- FORMATTED SMS PP UPD.

- UNFORMATTED SMS PP UPD.

- UNFORMATTED SMS PP ENV.

- FORMATTED SMS PP ENV.

- UNFORMATTED SMS CB.

Test Source Test\_Api\_2\_Ueh\_Gtpo.java

Test Applet: Api\_2\_Ueh\_Gtpo\_1.java

Cap File: Api\_2\_Ueh\_Gtpo.cap

5.2.2.5.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | 1 to 21. |
| N2 | 4, 10, 15, 20. |
| N3 | 1, 2, 3, 4, 5, 6 |
| N4 | 7, 8, 9, 10, 11 |
| N5 | 12, 13, 14, 15, 16 |
| N6 | 17, 18, 19, 20, 21 |
| N7 | 6 |
| C1 | 22 |
| C2 | Not testable |

5.2.2.5.4 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
|  | **FORMATTED SMS PP ENV triggering** |  |  |
| 1 | Test with TP-OA length of 2 | Returns 0x0D |  |
| 2 | Test with TP-OA length of 6 | Returns 0x0F |  |
| 3 | Test with TP-OA length of 12 | Returns 0x12 |  |
| 4 | Send a SMS PP with 2 TPDU TLV and inside two different UDL offsets | Returns 0x10 (the first offset) |  |
| 5 | Same test as 1, but with a concatenated SMS (2 Short Messages and maximum Secured Data Length = 0x00FA) | Returns 0x0D |  |
| 6 | Verify after call of the method the current TLV is the TPDU TLV:  findTLV device identities, getTPUDLOffset and then getValueByte to verify that the current TLV is the TPDU TLV | Returns 0x40 |  |
|  | **FORMATTED SMS PP UPD triggering** |  |  |
| 7 | Same test as 1 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x0D |  |
| 8 | Same test as 2 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x0F |  |
| 9 | Same test as 3 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x12 |  |
| 10 | Same test as 4 but with FORMATTED\_SMS\_PP\_UPD | Returns 0x10 (the first offset) |  |
| 11 | Same test as 7, but with a concatenated SMS (2 Short Messages and maximum Secured Data Length = 0x00FA) | Returns 0x0D |  |
|  | **UNFORMATTED SMS PP UPD triggering** |  |  |
| 12 | Same test as 1 but with UNFORMATTED\_SMS\_PP\_UPD | Returns 0x0D |  |
| 13 | Same test as 2 but with UNFORMATTED\_SMS\_PP\_UPD | Returns 0x0F |  |
| 14 | Same test as 3 but with UNFORMATTED\_SMS\_PP\_UPD | Returns 0x12 |  |
| 15 | Same test as 4 but with UNFORMATTED\_SMS\_PP\_UPD | Returns 0x12 (the first offset) |  |
| 16 | Same test as 12, but with a concatenated SMS (2 Short Messages and maximum User Data Length = 0x010C) | Returns 0x0D |  |
|  | **UNFORMATTED SMS PP ENV triggering** |  |  |
| 17 | Same test as 1 but with UNFORMATTED\_SMS\_PP\_ENV | Returns 0x0D |  |
| 18 | Same test as 2 but with UNFORMATTED\_SMS\_PP\_ENV | Returns 0x0F |  |
| 19 | Same test as 3 but with UNFORMATTED\_SMS\_PP\_ENV | Returns 0x12 |  |
| 20 | Same test as 4 but with UNFORMATTED\_SMS\_PP\_ENV | Returns 0x10 (the first offset) |  |
| 21 | Same test as 17, but with a concatenated SMS (2 Short Messages and maximum User Data Length = 0x010C) | Returns 0x0D |  |
|  | **UNFORMATTED SMS CB triggering** |  |  |
| 22 | Send an envelope Unformatted SMS CB, getTPUDLOffset | ToolkitException.UNAVAILABLE\_ELEMENT |  |

#### 5.2.2.6 Method getUserDataLength

Test Area Reference: Api\_2\_Ueh\_Gudl

5.2.2.6.1 Conformance requirement

The method with following header shall be compliant to its definition in the API.

public short getUserDataLength()

throws uicc.toolkit.ToolkitException

5.2.2.6.1.1 Normal execution

- CRRN1: The method shall return the length of the User Data contained in the SMS TPDU TLV element.

- CRRN2: The length is from the first SMS TPDU TLV element.

- CRRN3: If the SMS TPDU TLV element is available, it becomes the selected TLV

- CRRN4: The method can be used if the event is EVENT\_FORMATTED\_SMS\_PP\_ENV.

- CRRN5: The method can be used if the event is EVENT\_FORMATTED\_SMS\_PP\_UPD.

- CRRN6: The method can be used if the event is EVENT\_UNFORMATED\_SMS\_PP\_ENV.

- CRRN7: The method can be used if the event is EVENT\_UNFORMATTED\_SMS\_PP\_UDP.

5.2.2.6.1.2 Context errors

- CRRC1: The method shall throw UNAVAILABLE\_ELEMENT in case of unavailable TPDU TLV element.

- CRRC2: The method shall throw UNAVAILABLE\_ELEMENT in case of wrong data format.

5.2.2.6.2 Test area files

Specific triggering:

- FORMATTED SMS PP UPD.

- UNFORMATTED SMS PP UPD.

- FORMATED SMS PP ENV.

- UNFORMATED SMS PP ENV.

- For Formatted triggering if CC/RC/DS is used, the security parameters are the one defined in clause "5.3.8 Framework Security Management".

Test Source: Test\_Api\_2\_Ueh\_Gudl.java

Test Applet: Api\_2\_Ueh\_Gudl\_1.java

Cap File: Api\_2\_Ueh\_Gudl.cap

5.2.2.6.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | All test cases excepted:  53 |
| N2 | 11, 26, 37, 45 |
| N3 | 12, 27, 38, 46 |
| N4 | 1 to 15 |
| N5 | 16 to 30 |
| N6 | 31 to 38 |
| N7 | 39 to 46 |
| C1 | 47 |
| C2 | Not applicable |

5.2.2.6.4 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
|  | FORMATTED SMS PP ENV Triggering |  |  |
| 1 | Test with FORMATTED\_SMS\_PP\_ENV and TP-OA length of 2 and user data length of 0x3D | Returns 0x003D |  |
| 2 | Test with TP-OA length of 12 and user data length of 0x3D | Returns 0x003D |  |
| 3 | Test with RC/CC/DS length of 0 and secured data length of 0x10 | Returns 0x0023 |  |
| 4 | Test with RC/CC/DS length of 8 and secured data length of 0x10 | Returns 0x002B |  |
| 5 | Test with PCNTR = 0, no RC/CC/DS and data length of 0x10 | Returns 0x0023 |  |
| 6 | Test with PCNTR = 7, no RC/CC/DS and data length of 0x03 (ciphering shall be used) | Returns 0x001D |  |
| 7 | Test with SecuredDataLength = 00 and no RC/CC/DS | Returns 0x0013 |  |
| 8 | Test with UserDataLength = 0x7F | Returns 0x007F |  |
| 9 | Test with UserDataLength = 0x80 | Returns 0x0080 |  |
| 10 | Test with UserDataLength = maximum length (0x8C) for a single SMS | Returns 0x008C |  |
| 11 | Verify it is the first TPDU TLV:  Send a SMS PP with 2 TPDU TLV with two different user data lengths: 0x18 and 0x23 | Returns 0x0018 |  |
| 12 | Send envelope SMS-PP Formatted.  FindTLV() with TAG\_DEVICE\_IDENTITIES. GetUserDataLength() and then getValueByte() with offset 0 | GetValueByte() returns 0x40(TS 23.040 [11] first byte) |  |
| 13 | Test with UserDataLength = 0xFF with 2 concatenated SMS | Returns 0x00FF |  |
| 14 | Test with UserDataLength = 0x100 with 2 concatenated SMS | Returns 0x0100 |  |
| 15 | Test with UserDataLength = maximum length (0x010D) with 2 concatenated SMS | Returns 0x010D |  |
|  | FORMATTED SMS PP UPD Triggering |  |  |
| 16 | Test with FORMATTED\_SMS\_PP\_UPD and TP-OA length of 2 and user data length of 0x3D | Returns 0x003D |  |
| 17 | Test with TP-OA length of 12 and user data length of 0x3D | Returns 0x003D |  |
| 18 | Test with RC/CC/DS length of 0 and secured data length of 0x10 | Returns 0x0023 |  |
| 19 | Test with RC/CC/DS length of 8 and secured data length of 0x10 | Returns 0x002B |  |
| 20 | Test with PCNTR = 0, no RC/CC/DS and data length of 0x10 | Returns 0x0023 |  |
| 21 | Test with PCNTR = 7, no RC/CC/DS and data length of 0x03 (ciphering shall be used) | Returns 0x001D |  |
| 22 | Test with SecuredDataLength = 00 and no RC/CC/DS | Returns 0x0013 |  |
| 23 | Test with UserDataLength = 0x7F | Returns 0x007F |  |
| 24 | Test with UserDataLength = 0x80 | Returns 0x0080 |  |
| 25 | Test with UserDataLength = maximum length(0x8C) for a single SMS | Returns 0x008C |  |
| 26 | Verify it is the first TPDU TLV:  Send a SMS PP with 2 TPDU TLV with two different user data lengths: 0x18 and 0x23 | Returns 0x0018 |  |
| 27 | Send envelope SMS-PP Formatted.  FindTLV() with TAG\_DEVICE\_IDENTITIES. GetUserDataLength() and then getValueByte() with offset 0 | GetValueByte() returns 0x40(TS 23.040 [11] first byte) |  |
| 28 | Test with UserDataLength = 0xFF with 2 concatenated SMS | Returns 0x00FF |  |
| 29 | Test with UserDataLength = 0x100 with 2 concatenated SMS | Returns 0x0100 |  |
| 30 | Test with UserDataLength = maximum length (0x010D) with 2 concatenated SMS | Returns 0x010D |  |
|  | UNFORMATTED SMS PP ENV Triggering |  |  |
| 31 | Test with UNFORMATTED\_SMS\_PP\_ENV and TP-OA length of 2, and user data length of 0x3D | Returns 0x003D |  |
| 32 | Test with TP-OA length of 12, and user data length of 0x3D | Returns 0x003D |  |
| 33 | Test with UserDataLength = 0x00 | Returns 0x0000 |  |
| 34 | Test with UserDataLength = 0x7F | Returns 0x007F |  |
| 35 | Test with UserDataLength = 0x80 | Returns 0x0080 |  |
| 36 | Test with UserDataLength = maximum length: 0x8C for a single SMS | Returns 0x008C |  |
| 37 | Verify it is the first TPDU TLV:  Send a SMS PP with 2 TPDU TLV with two different user data lengths: 0x18 and 0x23 | Returns 0x0018 |  |
| 38 | Send envelope SMS-PP Unformatted.  FindTLV() with TAG\_DEVICE\_IDENTITIES. GetUserDataLength() and then getValueByte() with offset 0 (first user data = 0x55) | GetValueByte() returns 0x00  (TS 23.040 [11] first byte) |  |
|  | UNFORMATTED SMS PP UPD Triggering |  |  |
| 39 | Test with UNFORMATTED\_SMS\_PP\_UPD and TP-OA length of 2, and user data length of 0x3D | Returns 0x003D |  |
| 40 | Test with TP-OA length of 12, and user data length of 0x3D | Returns 0x003D |  |
| 41 | Test with UserDataLength = 0x00 | Returns 0x0000 |  |
| 42 | Test with UserDataLength = 0x7F | Returns 0x007F |  |
| 43 | Test with UserDataLength = 0x80 | Returns 0x0080 |  |
| 44 | Test with UserDataLength = maximum length: 0x8C for a single SMS | Returns 0x008C |  |
| 45 | Verify it is the first TPDU TLV:  Send a SMS PP with 2 TPDU TLV with two different user data lengths: 0x18 and 0x23 | Returns 0x0018 |  |
| 46 | Send envelope SMS-PP Unformatted.  FindTLV() with TAG\_DEVICE\_IDENTITIES. GetUserDataLength() and then getValueByte() with offset 0 | GetValueByte() returns 0x00  (TS 23.040 [11] first byte) |  |
|  | **UNRECOGNIZED\_ENVELOPE Triggering** |  |  |
| 47 | Test with an UNRECOGNIZED\_ENVELOPE | ToolkitException.UNAVAILABLE\_ELEMENT |  |

#### 5.2.2.7 Method getItemIdentifier

Test Area Reference: Api\_2\_Ueh\_Giid

5.2.2.7.1 Conformance requirement

The method with following header shall be compliant to its definition in the API.

public byte getItemIdentifier()

throws ToolkitException

5.2.2.7.1.1 Normal execution

- CRRN1:The method shall return the item identifier byte value.

- CRRN2:The item identifier byte value returned shall be from the first Item Identifier TLV element.

- CRRN3: If the element is available it becomes the TLV selected.

- CRRN4: The item identifier is available for all triggered toolkit applets from the invocation to the termination of their processToolkit method if the USATEnvelopeHandler is available.

5.2.2.7.1.2 Parameter Errors

No requirements.

5.2.2.7.1.3 Context errors

- CRRC1: The method shall throw ToolkitException.UNAVAILABLE\_ELEMENT if the item identifier TLV is not present.

- CRRC2: The method shall throw ToolkitException.OUT\_OF\_TLV\_BOUNDARIES if the item identifier byte is missing in the Item Identifier Comprehension TLV.

5.2.2.7.2 Test area files

Test Source: Test\_Api\_2\_Ueh\_Giid.java

Test Applet: Api\_2\_Ueh\_Giid\_1.java

Cap File: Api\_2\_Ueh\_Giid.cap

5.2.2.7.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | 1, 2, 3 |
| N2 | 2, 3 |
| N3 | 4 |
| N4 | 6 |
| C1 | 5 |
| C2 | 7 |

5.2.2.7.4 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| 1 | Send envelope formatted SMS with an item identifier TLV and identifier value of 03 | Returns 03 |  |
| 2 | Send envelope formatted SMS with two item identifier TLV with first value FF and second 44 | Returns FF |  |
| 3 | Send envelope formatted SMS with two item identifier TLV with first value 81 and second 44, call twice the method getItemIdentifier() | Returns 81  Returns 81 |  |
| 4 | Send envelope formatted SMS with item identifier TLV and value of 66. FindTLV() with TAG 02. getItemIdentifier() and then getValueByte() with offset 0 | getItemIdentifier()=getValueByte() |  |
| 5 | Send envelope formatted SMS without item identifier TLV and getItemIdentifier() | ToolkitException.UNAVAILABLE\_ELEMENT |  |
| 6 | Send envelope formatted SMS with item identifier TLV (66), send proactive command. Then getItemIdentifier() | Returns 66 |  |
| 7 | Send envelope formatted SMS with item identifier TLV but without item number | ToolkitException.OUT\_OF\_TLV\_BOUNDARIES |  |

#### 5.2.2.8 Method getChannelIdentifier

Test Area Reference: Api\_2\_Ueh\_Gcid

5.2.2.8.1 Conformance requirement

The method with following header shall be compliant to its definition in the API.

public byte getChannelIdentifier()

throws ToolkitException

5.2.2.8.1.1 Normal execution

- CRRN1: The method shall return the channel identifier byte value.

- CRRN2: The channel identifier byte value returned shall be from the first Channel status TLV element.

- CRRN3: If the element is available it becomes the currently selected TLV.

- CRRN4: The channel identifier is available for all triggered toolkit applets from the invocation to the termination of their processToolkit method if the USATEnvelopeHandler is available.

5.2.2.8.1.2 Context errors

- CRRC1: The method shall throw ToolkitException.UNAVAILABLE\_ELEMENT if the Channel status TLV is not present.

- CRRC2: The method shall throw ToolkitException.OUT\_OF\_TLV\_BOUNDARIES if the Comprehension TLV Channel Status length is equal to 0.

5.2.2.8.2 Test area files

Test Source: Test\_Api\_2\_Ueh\_Gcid.java

Test Applet: Api\_2\_Ueh\_Gcid\_1.java

Cap File: Api\_2\_Ueh\_Gcid.cap

5.2.2.8.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | 1, 2 |
| N2 | 3 |
| N3 | 3 |
| N4 | 5 |
| C1 | 4 |
| C2 | 6 |

5.2.2.8.4 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| 0 | 1- Applet1 is installed with maximum number of channel = 07.  2- Applet1 builds proactive commands OPEN CHANNEL with init() method in order to open all channels.  ProactiveHandler.send() method is called. |  | 2- OPEN CHANNEL proactive command is fetched  TERMINAL RESPONSE is issued with Channel Id from 01 to 07 |
| 1 | **Successful Call**  1- Send envelope Event Download Channel Status with channel status TLV:  channel status value = 0x8100.  2- Call USATEnvelopeHandler.getChannelIdentifier() method | 1- Applet1 is triggered  2- Returns 0x01 |  |
| 2 | **Two channel status elements**  1- Send envelope Event Download Channel Status with two channel status TLV:  first value = 0x8400  second value = 0x8500.  2- Call twice the USATEnvelopeHandler.getChannelIdentifier() method | 2- Returns twice 0x04 |  |
| 3 | **Verify current TLV**  1- Send envelope Event Download Channel Status with channel status TLV:  Channel Status value = 0x0605  ViewHandler.FindTLV() with Device Identity Tag.  2- Call USATEnvelopeHandler.getChannelIdentifier() method.  3- Compare USATEnvelopeHandler.getChannelIdentifier() and then ViewHandler.getValueByte(0). | 2- Returns 0x06  3- GetChannelIdentifier() =getValueByte(0) |  |
| 4 | **UNAVAILABLE\_ELEMENT exception**  1- Send envelope Menu Selection without Channel Status TLV.  2- Call USATEnvelopeHandler.getChannelIdentifier() method. | 2- A Toolkit exception UNAVAILABLE\_ELEMENT is thrown. |  |
| 5 | **Successful Call**  1- Send Envelope Event Download Channel Status with Channel Status TLV:  Channel status value = 0x0600  2- Call USATEnvelopeHandler.getChannelIdentifier() method. | 1- Returns 0x06 |  |
| 6 | **OUT\_OF\_TLV\_BOUNDARIES exception**  1- Send unrecognized envelope with a Channel Status TLV having a length equal to 0.  2- Call USATEnvelopeHandler.getChannelIdentifier() method. | 2- A Toolkit exception OUT\_OF\_TLV\_BOUNDARIES is thrown. |  |

#### 5.2.2.9 Method getChannelStatus

Test Area Reference: Api\_2\_Ueh\_Gcst

5.2.2.9.1 Conformance requirement

The method with following header shall be compliant to its definition in the API.

public short getChannelStatus(byte channelIdentifier)

throws ToolkitException

5.2.2.9.1.1 Normal execution

- CRRN1: The method shall return the value of the first Channel Status TLV element whose channel identifier is equal to the channelIdentifier parameter.

- CRRN2: The Channel Status value returned shall be from the element whose channel identifier is equal to the ChannelIdentifier parameter.

- CRRN3: If the element is available it becomes the currently selected TLV.

- CRRN4: The channel status is available for all triggered toolkit applets from the invocation to the termination of their processToolkit method if the USATEnvelopeHandler is available.

5.2.2.9.1.2 Context errors

- CRRC1: The method shall throw ToolkitException.UNAVAILABLE\_ELEMENT if no Channel Status TLV element with the right identifier could be found.

- CRRC2: The method shall throw ToolkitException.OUT\_OF\_TLV\_BOUNDARIES if a Channel Status TLV element with the right identifier could be found but its value is less than 2 bytes long.

5.2.2.9.2 Test area files

Test Source: Test\_Api\_2\_Ueh\_Gcst.java

Test Applet: Api\_2\_Ueh\_Gcst\_1.java

Cap File: Api\_2\_Ueh\_Gcst.cap

5.2.2.9.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | 6 |
| N2 | 5, 7 |
| N3 | 8 |
| N4 | 9 |
| C1 | 1, 2 |
| C2 | 3, 4 |

5.2.2.9.4 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| 0 | 1- Applet1 is installed with maximum number of channel = 01.  2- Applet1 builds proactive commands OPEN CHANNEL with init() method in order to open a channel.  ProactiveHandler.send() method is called. |  | 2- OPEN CHANNEL proactive command is fetched  TERMINAL RESPONSE is issued with channel status value = 0x8100 |
| 1 | Channel status TLV is not present  1- Send envelope unrecognized  2- Call USATEnvelopeHandler.getChannelStatus(0x01) method. | 2- UNAVAILABLE\_ELEMENT ToolkitException is thrown |  |
| 2 | Channel status TLV with the identifier is not present  1- Send envelope Event Download Channel Status with Channel status Value = 0x8100  2- Call USATEnvelopeHandler.getChannelStatus(0x02) method. | 2- UNAVAILABLE\_ELEMENT ToolkitException is thrown |  |
| 3 | **Channel status TLV with a length equal to 0**  1- Send envelope unrecognized with Channel status TLV with a length equal to 0.  2- Call USATEnvelopeHandler.getChannelStatus(0x01) method. | 2- UNAVAILABLE\_ELEMENT ToolkitException is thrown |  |
| 4 | **Channel status TLV with a length equal to 1**  1- - Send envelope unrecognized with Channel status TLV with a length equal to 1.  2- Call USATEnvelopeHandler.getChannelStatus(0x01) method. | 2- OUT\_OF\_TLV\_BOUNDARIES ToolkitException is thrown |  |
| 5 | **Get channel status value**  1- Send envelope Event Download Channel Status with Channel status value=0x8100.  2- Call USATEnvelopeHandler.getChannelStatus(0x01) method. | 2- Returns 0x8100 |  |
| 6 | Get channel status value with 2 TLV  1- Send envelope Event Download Channel Status with 2 channel status value: 0x8100 and 0x8101.  2- Call USATEnvelopeHandler.getChannelStatus(0x01) method. | 2- Returns 0x8100 |  |
| 7 | Get channel status value with 2 TLV  1- Send envelope Event Download Channel Status with 2 channel status value: 0x8100 and 0x8201.  2- Call USATEnvelopeHandler.getChannelStatus(0x02) method. | 2- Returns 0x8201 |  |
| 8 | Channel status TLV is currently selected TLV  1- Send envelope Event Download Channel Status with channel status value 0x8100.  Call ViewHandler.FindTLV() method with Device Identity Tag.  2- Call USATEnvelopeHandler.getChannelStatus(0x01) method.  3- Compare USATEnvelopeHandler.getChannelStatus(0x01) and ViewHandler.getValueShort(0) method results. | 2- Returns 0x8100  3- Check getChannelStatus() =getValueShort(0) |  |
| 9 | **Get channel status value after a proactive command**  1- Send envelope Event Download Channel Status with Channel status value=0x8100.  2- Call USATEnvelopeHandler.getChannelStatus(0x01) method.  3- Send a proactive command display text  4- Call USATEnvelopeHandler.getChannelStatus(0x01) method. | 2- Returns 0x8100  4- Returns 0x8100 | 3- DISPLAY TEXT proactive command is fetched  TERMINAL RESPONSE is issued |

#### 5.2.2.10 Method getSize

Test Area Reference: Api\_2\_Ueh\_Gtsz

5.2.2.10.1 Conformance requirement

The method with following header shall be compliant to its definition in the API.

public short getSize()

5.2.2.10.1.1 Normal execution

- CRRN1: Returns the BER TLV size, this includes the tag and the length.

5.2.2.10.1.2 Parameter errors

No requirements

5.2.2.10.1.3 Context errors

No requirements

5.2.2.10.2 Test area files

Specific triggering: Unrecognized Envelope

Test Source: Test\_Api\_2\_Ueh\_Gtsz.java

Test Applet: Api\_2\_Ueh\_Gtsz\_1.java

Cap File: Api\_2\_Ueh\_Gtsz.cap

5.2.2.10.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| 1 | 1, 2 |

5.2.2.10.4 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| 0 | Send an unrecognized envelope of length 0x33 (including tag and length) |  |  |
| 1 | Call getSize() method just after triggering of the application. | Returns 0x33 |  |
| 2 | Call getSize() method after a proactive command. | Returns 0x33 |  |

#### 5.2.2.11 Method getTag

Test Area Reference: Api\_2\_Ueh\_Gttg

5.2.2.11.1 Conformance requirement

The method with following header shall be compliant to its definition in the API.

public short getTag()

5.2.2.11.1.1 Normal execution

- CRRN1: Returns the BER Tag of the BER TLV list.

5.2.2.11.1.2 Parameter errors

No requirements.

5.2.2.11.1.3 Context errors

No requirements.

5.2.2.11.2 Test area files

Specific triggering: Unrecognized Envelope

Test Source: Test\_Api\_2\_Ueh\_Gttg.java

Test Applet: Api\_2\_Ueh\_Gttg\_1.java

Cap File: Api\_2\_Ueh\_Gttg.cap

5.2.2.11.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| 1 | 1, 2 |

5.2.2.11.4 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| 0 | Send an unrecognized envelope |  |  |
| 1 | Call getTag() method just after triggering of the application. | Returns 0x01 |  |
| 2 | Call getTag() method after a proactive command. | Returns 0x01 |  |

#### 5.2.2.12 Method compareValue

Test Area Reference: Api\_2\_Ueh\_Cprv

5.2.2.12.1 Conformance requirement

The method with following header shall be compliant to its definition in the API.

public byte compareValue(short valueOffset,

byte[] compareBuffer,

short compareOffset,

short compareLength)

throws java.lang.NullPointerException,

java.lang.ArrayIndexOutOfBoundsException,

ToolkitException

5.2.2.12.1.1 Normal execution

Compares the last found TLV element with a buffer:

- CRRN1: returns 0 if identical.

- CRRN2: returns -1 if the first miscomparing byte in Comprehension TLV List is less than that in compareBuffer.

- CRRN3: returns 1 if the first miscomparing byte in Comprehension TLV List is greater than that in compareBuffer.

5.2.2.12.1.2 Parameter errors

- CRRP1: if compareBuffer is null NullPointerException shall be thrown.

- CRRP2: if compareOffset or compareLength or both would cause access outside array bounds, or if compareLength is negative ArrayIndexOutOfBoundsException shall be thrown.

- CRRP3: if valueOffset, dstLength or both are out of the current TLV an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException.OUT\_OF\_TLV\_BOUNDARIES.

5.2.2.12.1.3 Context errors

- CRRC1: if the handler is busy an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException.HANDLER\_NOT\_AVAILABLE.

- CRRC2: in case of unavailable TLV element an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException.UNAVAILABLE\_ELEMENT.

5.2.2.12.2 Test area files

Specific triggering: Unrecognized Envelope

Test Source: Test\_Api\_2\_Ueh\_Cprv.java

Test Applet: Api\_2\_Ueh\_Cprv\_1.java

Cap File: Api\_2\_Ueh\_Cprv.cap

5.2.2.12.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | 12, 15 |
| N2 | 13, 16, 18 |
| N3 | 14, 17 |
| P1 | 1 |
| P2 | 2, 3, 4, 5, 6 |
| P3 | 7, 8, 9, 10 |
| C1 | Does not apply for USATEnvelopeHandler |
| C2 | 11 |

5.2.2.12.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
| 1 | Search TLV 02h |  |  |
|  | compareValue() with a null compareBuffer | NullPointerException is thrown |  |
| 2 | Search TLV 0Bh |  |  |
|  | compareOffset ≥ compareBuffer.length  compareValue()  compareBuffer.length = 5  compareOffset = 5  compareLength = 1 | ArrayIndexOutOfBoundsException is thrown |  |
| 3 | compareOffset < 0  compareValue()  compareBuffer.length = 5  compareOffset = -1  compareLength = 1 | ArrayIndexOutOfBoundsException is thrown |  |
| 4 | compareLength >compareBuffer.length  compareValue()  compareBuffer.length = 5  compareOffset = 0  compareLength = 6 | ArrayIndexOutOfBoundsException is thrown |  |
| 5 | compareOffset + compareLength >compareBuffer.length  compareValue()  compareBuffer.length = 5  compareOffset = 3  compareLength = 3 | ArrayIndexOutOfBoundsException is thrown |  |
| 6 | compareLength < 0  compareValue()  compareBuffer.length = 5  compareOffset = 0  compareLength = -1 | ArrayIndexOutOfBoundsException is thrown |  |
| 7 | Search TLV 06h |  |  |
|  | valueOffset ≥ TLV Length  compareValue()  valueOffset = 6  compareBuffer.length = 15  compareOffset = 0  compareLength = 1 | ToolkitException.OUT\_OF\_TLV\_BOUNDARIES is thrown |  |
| 8 | valueOffset < 0  compareValue()  valueOffset = -1  compareBuffer.length = 15  compareOffset = 0  compareLength = 1 | ToolkitException.OUT\_OF\_TLV\_BOUNDARIES is thrown |  |
| 9 | compareLength > TLV length  compareValue()  valueOffset = 0  compareBuffer.length = 15  compareOffset = 0  compareLength = 7 | ToolkitException.OUT\_OF\_TLV\_BOUNDARIES is thrown |  |
| 10 | valueOffset + compareLength > TLV length  compareValue()  valueOffset = 2  compareBuffer.length = 15  compareOffset = 0  compareLength = 5 | ToolkitException.OUT\_OF\_TLV\_BOUNDARIES is thrown |  |
| 11 | Search TLV 01h | Result is TLV\_NOT\_FOUND |  |
|  | compareValue() | ToolkitException.UNAVAILABLE\_ELEMENT is thrown |  |
| 12 | Search TLV 06h |  |  |
|  | Initialize compareBuffer  compareBuffer =  81 11 22 33 44 F5 |  |  |
|  | Compare buffers  compareValue()  valueOffset = 0  compareOffset = 0  compareLength = 6 | Result is 00h |  |
| 13 | Initialize compareBuffer  compareBuffer =  7F 11 22 33 44 F5 |  |  |
|  | Compare buffers with same parameters | Result is -1 |  |
| 14 | Initialize compareBuffer  compareBuffer =  83 11 22 33 44 F5 |  |  |
|  | Compare buffers with same parameters | Result is -1 |  |
| 15 | Initialize compareBuffer  compareBuffer =  55 55 55 81 11 22 33 44 F5  55 55 55 55 55 |  |  |
|  | Compare buffers  compareValue()  valueOffset = 1  compareOffset = 4  compareLength = 5 | Result is 00h |  |
| 16 | Initialize compareBuffer  compareBuffer =  55 55 55 81 10 22 33 44 F5  55 55 55 55 55 |  |  |
|  | Compare buffers with same parameters | Result is +1 |  |
| 17 | Initialize compareBuffer  compareBuffer =  55 55 55 81 12 22 33 44 F5  55 55 55 55 55 |  |  |
|  | Compare buffers with same parameters | Result is -1 |  |
| 18 | **Successful call, compareValue() with length=0**  CompareBuffer.length = 15  CompareOffset = 15  CompareLength = 0 | Result of compareValue() is 0 |  |

#### 5.2.2.13 Method copy

Test Area Reference: Api\_2\_Ueh\_Copy

5.2.2.13.1 Conformance requirement

The method with following header shall be compliant to its definition in the API.

public short copy(byte[] dstBuffer,

short dstOffset,

short dstLength)

throws java.lang.NullPointerException,

java.lang.ArrayIndexOutOfBoundsException,

ToolkitException

5.2.2.13.1.1 Normal execution

- CRRN1: copies the Comprehension TLV list contained in the handler to the destination byte array.

- CRRN2: returns dstOffset + dstLength.

5.2.2.13.1.2 Parameter errors

- CRRP1: if dstBuffer is null a NullPointerException is thrown.

- CRRP2: if dstOffset or dstLength or both would cause access outside array bounds, or if dstLength is negative, an ArrayIndexOutOfBoundsException is thrown.

- CRRP3: if dstLength is greater than the length of the Comprehension TLV List, an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException. OUT\_OF\_TLV\_BOUNDARIES.

5.2.2.13.1.3 Context errors

- CRRC1: if the handler is busy an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException.HANDLER\_NOT\_AVAILABLE.

5.2.2.13.2 Test area files

Specific triggering: Unrecognized Envelope

Test Source: Test\_Api\_2\_Ueh\_Copy.java

Test Applet: Api\_2\_Ueh\_Copy\_1.java

Cap File: Api\_2\_Ueh\_Copy.cap

5.2.2.13.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | 9, 11, 13, 15 |
| N2 | 8, 10, 12, 14, 16 |
| P1 | 1 |
| P2 | 2, 3, 4, 5, 6 |
| P3 | 7 |
| C1 | Does not apply for USATEnvelopeHandler |

5.2.2.13.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
| 1 | NULL as parameter to dstBuffer | NullPointerException is thrown |  |
| 2 | dstOffset ≥ dstBuffer.length  copy()  dstBuffer.length = 5  dstOffset = 5  dstLength = 1 | ArrayIndexOutOfBoundsException is thrown |  |
| 3 | dstOffset < 0  copy()  dstBuffer.length = 5  dstOffset = -1  dstLength = 1 | ArrayIndexOutOfBoundsException is thrown |  |
| 4 | dstLength > dstBuffer.length  copy()  dstBuffer.length = 5  dstOffset = 0  dstLength = 6 | ArrayIndexOutOfBoundsException is thrown |  |
| 5 | DstOffset + dstLength > dstBuffer.length  copy()  DstBuffer.length = 5  DstOffset = 3  DstLength = 3 | ArrayIndexOutOfBoundsException is thrown |  |
| 6 | dstLength < 0  copy()  dstBuffer.length = 5  dstOffset = 0  dstLength = -1 | ArrayIndexOutOfBoundsException is thrown |  |
| 7 | DstLength > length of the Comprehension TLV list  copy()  DstBuffer.length = 48  DstOffset = 0  DstLength = 48 | ToolkitException.OUT\_OF\_TLV\_BOUNDARIES is thrown |  |
| 8 | Successful call, dstBuffer is the whole buffer  copy()  DstBuffer.length = 47  DstOffset = 0  DstLength = 47 | Result of copy() is 0X0047 |  |
| 9 | Compare the buffer | Result of arrayCompare() is 0 |  |
| 10 | Successful call, dstBuffer is part of a buffer  copy()  DstBuffer.length = 50  dstOffset = 3  dstLength = 47 | Result of copy() is 0X0032 |  |
| 11 | Compare the whole buffer | Result of arrayCompare() is 0 |  |
| 12 | Successful call, dstBuffer is part of a buffer  copy()  dstBuffer.length = 15  dstOffset = 3  dstLength = 6 | Result of copy() is 0X0009 |  |
| 13 | Compare the whole buffer | Result of arrayCompare() is 0 |  |
| 14 | **Successful call, dstBuffer is part of a buffer**  copy()  dstBuffer.length = 260  dstOffset = 257  dstLength = 3 | Result of copy() is 0X0104 |  |
| 15 | Compare the whole buffer | Result of arrayCompare() is 0 |  |
| 16 | Successful call, copy() with length =0  dstBuffer.length = 260  dstOffset = 260  dstLength = 0 | Result of copy() is 0x104 |  |

#### 5.2.2.14 Method copyValue

Test Area Reference: Api\_2\_Ueh\_Cpyv

5.2.2.14.1 Conformance requirement

The method with following header shall be compliant with its definition in the API.

public short copyValue(short valueOffset,

byte[] dstBuffer,

short dstOffset,

short dstLength)

throws java.lang.NullPointerException,

java.lang.ArrayIndexOutOfBoundsException,

ToolkitException

5.2.2.14.1.1 Normal execution

- CRRN1: copies a part of the last TLV element which has been found, into a destination. buffer.

- CRRN2: returns dstOffset + dstLength.

5.2.2.14.1.2 Parameter errors

- CRRP1: if dstBuffer is null NullPointerException is thrown.

- CRRP2: if dstOffset or dstLength or both would cause access outside array bounds, or if dstLength is negative ArrayIndexOutOfBoundsException is thrown.

- CRRP3: if valueOffset, dstLength or both are out of the current TLV an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException OUT\_OF\_TLV\_BOUNDARIES.

5.2.2.14.1.3 Context errors

- CRRC1: if the handler is busy an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException.HANDLER\_NOT\_AVAILABLE.

- CRRC2: in case of unavailable TLV element an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException.UNAVAILABLE\_ELEMENT.

5.2.2.14.2 Test area files

Specific triggering: Unrecognized Envelope

Test Source: Test\_Api\_2\_Ueh\_Cpyv.java

Test Applet: Api\_2\_Ueh\_Cpyv\_1.java

Cap File: Api\_2\_Ueh\_Cpyv.cap

5.2.2.14.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | 13, 15 |
| N2 | 12, 14, 16 |
| P1 | 1 |
| P2 | 2, 3, 4, 5, 6 |
| P3 | 7, 8, 9, 10 |
| C1 | Does not apply for USATEnvelopeHandler |
| C2 | 11 |

5.2.2.14.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
| 1 | Search TLV 02h |  |  |
|  | copyValue() with a null dstBuffer | NullPointerException is thrown |  |
| 2 | Search TLV 0Bh |  |  |
|  | dstOffset ≥ dstBuffer.length  copyValue()  dstBuffer.length = 5  dstOffset = 5  dstLength = 1 | ArrayIndexOutOfBoundsException is thrown |  |
| 3 | dstOffset < 0  copyValue()  dstBuffer.length = 5  dstOffset = -1  dstLength = 1 | ArrayIndexOutOfBoundsException is thrown |  |
| 4 | dstLength >dstBuffer.length  copyValue()  dstBuffer.length = 5  dstOffset = 0  dstLength = 6 | ArrayIndexOutOfBoundsException is thrown |  |
| 5 | dstOffset + dstLength >dstBuffer.length  copyValue()  dstBuffer.length = 5  dstOffset = 3  dstLength = 3 | ArrayIndexOutOfBoundsException is thrown |  |
| 6 | dstLength < 0  copyValue()  dstBuffer.length = 5  dstOffset = 0  dstLength = -1 | ArrayIndexOutOfBoundsException is thrown |  |
| 7 | Search TLV 06h |  |  |
|  | valueOffset ≥ TLV Length  copyValue()  valueOffset = 6  dstBuffer.length = 15  dstOffset = 0  dstLength = 1 | ToolkitException.OUT\_OF\_TLV\_BOUNDARIES is thrown |  |
| 8 | valueOffset < 0  copyValue()  valueOffset = -1  dstBuffer.length = 15  dstOffset = 0  dstLength = 1 | ToolkitException.OUT\_OF\_TLV\_BOUNDARIES is thrown |  |
| 9 | dstLength > TLV length  copyValue()  valueOffset = 0  dstBuffer.length = 15  dstOffset = 0  dstLength = 7 | ToolkitException.OUT\_OF\_TLV\_BOUNDARIES is thrown |  |
| 10 | valueOffset + dstLength > TLV length  copyValue()  valueOffset = 2  dstBuffer.length = 15  dstOffset = 0  dstLength = 5 | ToolkitException.OUT\_OF\_TLV\_BOUNDARIES is thrown |  |
| 11 | Search TLV 01h |  |  |
|  | copyValue() | ToolkitException.UNAVAILABLE\_ELEMENT is thrown on the copyValue() method call. |  |
| 12 | Search TLV 06h |  |  |
|  | Successful call  copyValue()  valueOffset = 0  dstBuffer.length = 6  dstOffset = 0  dstLength = 6 | Result of copyValue() is 0x0006 |  |
| 13 | Compare buffer  buffer = 81 11 22 33 44 F5 | Result is 00h |  |
| 14 | initialize dstBuffer  dstBuffer = 55 55 … 55 |  |  |
|  | Successful call  copyValue()  valueOffset = 1  dstBuffer.length = 20  dstOffset = 3  dstLength = 4 | Result of copyValue() is 0x0007 |  |
| 15 | Compare buffer  buffer =  55 55 55 11 22  33 44 55 55 55  55 55 55 55 55  55 55 55 55 55 | Result is 00h |  |
| 16 | Successful call, copy with length =0  dstBuffer.length = 20  dstOffset = 20  dstLength = 0 | Result of copyValue() is 20 |  |

#### 5.2.2.15 Method findAndCompareValue(byte tag, byte[] compareBuffer, short compareOffset)

Test Area Reference: Api\_2\_Ueh\_Facrb\_Bs

5.2.2.15.1 Conformance requirement

The method with following header shall be compliant to its definition in the API.

public byte findAndCompareValue(byte tag,

byte[] compareBuffer,

short compareOffset)

throws java.lang.NullPointerException,

java.lang.ArrayIndexOutOfBoundsException,

ToolkitException

5.2.2.15.1.1 Normal execution

Looks for the first occurrence of a TLV element from beginning of a TLV list and compare its value with a buffer:

- CRRN1: if no TLV element is found, the UNAVAILABLE\_ELEMENT exception is thrown and the current TLV is no longer defined.

- CRRN2: if the method is successful then the corresponding TLV becomes current.

- CRRN3: if identical returns 0.

- CRRN4: if the first miscomparing byte in Comprehension TLV is less than that in compareBuffer returns -1.

- CRRN5: if the first miscomparing byte in Comprehension TLV is greater than that in compareBuffer returns 1.

- CRRN6: The search method is comprehension required flag independent.

5.2.2.15.1.2 Parameter errors

- CRRP1: if compareBuffer is null NullPointerException shall be thrown.

- CRRP2: if compareOffset would cause access outside array bounds ArrayIndexOutOfBoundsException shall be thrown.

5.2.2.15.1.3 Context errors

- CRRC1: if the handler is busy an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException.HANDLER\_NOT\_AVAILABLE.

5.2.2.15.2 Test area files

Specific triggering: Unrecognized Envelope

Test Source: Test\_Api\_2\_Ueh\_Facrb\_Bs.java

Test Applet: Api\_2\_Ueh\_Facrb\_Bs\_1.java

Cap File: Api\_2\_Ueh\_Facrb\_Bs.cap

5.2.2.15.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | 6,7 |
| N2 | 9 |
| N3 | 8, 12, 13 |
| N4 | 10, 14 |
| N5 | 11, 15 |
| N6 | 16, 17 |
| P1 | 1 |
| P2 | 2, 3, 4, 5 |
| C1 | Does not apply for USATEnvelopeHandler |

5.2.2.15.4 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
|  | Fill the Unrecognized Envelope with TLV: Tag 02, Value 83 81, Tag 06, Value 81 11 22 33 44 F5, Tag 02 Value 22 44 Tag 33, Length C4 Value 01 02 … |  |  |
| 1 | findAndCompareValue() with a null dstBuffer | NullPointerException is thrown |  |
| 2 | compareOffset ≥ compareBuffer.length  findAndCompareValue()  tag = 06h  compareBuffer.length = 12  compareOffset = 12 | ArrayIndexOutOfBoundsException is thrown |  |
| 3 | compareOffset < 0  findAndCompareValue()  compareBuffer.length = 12  compareOffset = -1 | ArrayIndexOutOfBoundsException is thrown |  |
| 4 | length > compareBuffer.length  findAndCompareValue()  compareBuffer.length = 05  compareOffset = 0 | ArrayIndexOutOfBoundsException is thrown |  |
| 5 | compareOffset + length > compareBuffer.length  findAndCompareValue()  compareBuffer.length = 12  compareOffset = 7 | ArrayIndexOutOfBoundsException is thrown |  |
| 6 | Select a TLV (tag 02h) |  |  |
|  | findAndCompareValue()  tag = 03h | ToolkitException.UNAVAILABLE\_ELEMENT is thrown |  |
| 7 | Call the getValueLength() method | ToolkitException.UNAVAILABLE\_ELEMENT is thrown. |  |
| 8 | Initialize compareBuffer  compareBuffer = 81 11 22 33 44 F5 |  |  |
|  | Compare buffers  findAndCompareValue()  tag = 06h  compareOffset = 0 | Result is 00h |  |
| 9 | Verify current TLV  getValueLength() | Result is 06 |  |
| 10 | Initialize compareBuffer  compareBuffer = 81 11 22 33 44 F4 |  |  |
|  | Compare buffers with same parameters | Result is +1 |  |
| 11 | Initialize compareBuffer  compareBuffer = 81 11 22 33 44 F6 |  |  |
|  | Compare buffers with same parameters | Result is -1 |  |
| 12 | Initialize compareBuffer  compareBuffer =  55 55 81 11 22 33 44 F5 55 55 55 55 |  |  |
|  | Compare buffers  findAndCompareValue()  compareOffset = 2 | Result is 00h |  |
| 13 | Initialize compareBuffer  compareBuffer =  55 55 83 81 55 55 55 55 55 55 55 55 |  |  |
|  | Compare buffers  findAndCompareValue()  compareOffset = 2 | Result is 00h |  |
| 14 | Initialize compareBuffer  compareBuffer =  55 55 83 80 55 55 55 55 55 55 55 55 |  |  |
|  | Compare buffers  findAndCompareValue()  compareOffset = 2 | Result is +1 |  |
| 15 | Initialize compareBuffer  compareBuffer =  55 55 83 82 55 55 55 55 55 55 55 55 |  |  |
|  | Compare buffers  findAndCompareValue()  compareOffset = 2 | Result is -1 |  |
| 16 | Initialize compareBuffer  compareBuffer =  83 81 55 55 55 55 55 55 55 55 55 55 |  |  |
|  | Successful call (with tag 02h)  findAndCompareValue()  tag = 02h  compareBuffer.length = 12  compareOffset = 0 | Result is 00h |  |
| 17 | Initialize compareBuffer  CompareBuffer = 01 02 … C4 |  |  |
|  | Successful call (with tag B3h)  findAndCompareValue()  Tag = B3h  CompareBuffer.length = C4  CompareOffset = 0 | Result is 00h |  |

#### 5.2.2.16 Method findAndCompareValue(byte tag, byte occurrence, short valueOffset, byte[] compareBuffer, short compareOffset, short compareLength)

Test Area Reference: Api\_2\_Ueh\_Facrbbs\_Bss

5.2.2.16.1 Conformance requirement

The method with following header shall be compliant to its definition in the API.

public byte findAndCompareValue(byte tag,

byte occurrence,

short valueOffset,

byte[] compareBuffer,

short compareOffset,

short compareLength)

throws java.lang.NullPointerException,

java.lang.ArrayIndexOutOfBoundsException,

ToolkitException

5.2.2.16.1.1 Normal execution

Looks for the indicated occurrence of a TLV element from the beginning of a TLV list and compare its value with a buffer:

- CRRN1: if no TLV element is found, the UNAVAILABLE\_ELEMENT exception is thrown and the current TLV is no longer defined.

- CRRN2: if the method is successful then the corresponding TLV becomes current.

- CRRN3: if identical 0 is returned.

- CRRN4: if the first miscomparing byte in Comprehension TLV is less than that in compareBuffer -1 is returned.

- CRRN5: if the first miscomparing byte in Comprehension TLV is greater than that in compareBuffer 1 is returned

- CRRN6: The search method is comprehension required flag independent.

5.2.2.16.1.2 Parameter errors

- CRRP1: if compareBuffer is null NullPointerException shall be thrown.

- CRRP2: if compareOffset or compareLength or both would cause access outside array bounds, or if compareLength is negative ArrayIndexOutOfBoundsException shall be thrown.

- CRRP3: if valueOffset, compareLength or both are out of the current TLV an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException.OUT\_OF\_TLV\_BOUNDARIES.

- CRRP4: if an input parameter is not valid (e.g. occurrence = 0) an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException.BAD\_INPUT\_PARAMETER.

5.2.2.16.1.3 Context errors

- CRRC1: if the handler is busy an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException.HANDLER\_NOT\_AVAILABLE.

5.2.2.16.2 Test area files

Specific triggering: Unrecognized Envelope

Test Source: Test\_Api\_2\_Ueh\_Facrbbs\_Bss.java

Test Applet: Api\_2\_Ueh\_Facrbbs\_Bss\_1.java

Cap File: Api\_2\_Ueh\_Facrbbs\_Bss.cap

5.2.2.16.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | 12, 13 |
| N2 | 15 |
| N3 | 14, 18, 21, 22, 26 |
| N4 | 16, 20 |
| N5 | 17, 19, 23 |
| N6 | 24, 25 |
| P1 | 1 |
| P2 | 2, 3, 4, 5, 6 |
| P3 | 7, 8, 9, 10 |
| P4 | 11 |
| C1 | Does not apply for USATEnvelopeHandler |

5.2.2.16.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
|  | Fill the Unrecognized Envelope with TLV: Tag 02, Value 83 81, Tag 06, Value 81 11 22 33 44 F5, Tag 02 Value 22 44 Tag 33, Length C4 Value 01 02 … |  |  |
| 1 | findAndCompareValue() with a null compareBuffer | NullPointerException is thrown |  |
| 2 | compareOffset ≥ compareBuffer.length  findAndCompareValue()  tag = 06h, occurrence = 1  valueOffset = 0  compareBuffer.length = 6  compareOffset = 6  compareLength = 1 | ArrayIndexOutOfBoundsException is thrown |  |
| 3 | compareOffset < 0  findAndCompareValue()  compareBuffer.length = 6  compareOffset = -1  compareLength = 1 | ArrayIndexOutOfBoundsException is thrown |  |
| 4 | compareLength >compareBuffer.length  findAndCompareValue()  compareBuffer.length = 5  compareOffset = 0  compareLength = 6 | ArrayIndexOutOfBoundsException is thrown |  |
| 5 | compareOffset + compareLength >compareBuffer.length  findAndCompareValue()  compareBuffer.length = 5  compareOffset = 3  compareLength = 3 | ArrayIndexOutOfBoundsException is thrown |  |
| 6 | compareLength < 0  findAndCompareValue()  compareBuffer.length = 5  compareOffset = 0  compareLength = -1 | ArrayIndexOutOfBoundsException is thrown |  |
| 7 | valueOffset ≥ Value Length  findAndCompareValue()  tag = 06h, occurrence = 1  valueOffset = 6  compareBuffer.length = 15  compareOffset = 0  compareLength = 1 | ToolkitException.OUT\_OF\_TLV\_BOUNDARIES is thrown |  |
| 8 | valueOffset < 0  findAndCompareValue()  valueOffset = -1  compareBuffer.length = 15  compareOffset = 0  compareLength = 1 | ToolkitException.OUT\_OF\_TLV\_BOUNDARIES is thrown |  |
| 9 | compareLength > Value length  findAndCompareValue()  valueOffset = 0  compareBuffer.length = 15  compareOffset = 0  compareLength = 7 | ToolkitException.OUT\_OF\_TLV\_BOUNDARIES is thrown |  |
| 10 | valueOffset + compareLength > Value length  findAndCompareValue()  valueOffset = 2  compareBuffer.length = 15  compareOffset = 0  compareLength = 5 | ToolkitException.OUT\_OF\_TLV\_BOUNDARIES is thrown |  |
| 11 | Invalid parameter  findAndCompareValue()  occurrence = 0 | ToolkitException.BAD\_INPUT\_PARAMETER is thrown |  |
| 12 | Select a TLV (tag 02h) |  |  |
|  | findAndCompareValue()  tag = 06h  occurrence = 2 | ToolkitException.UNAVAILABLE\_ELEMENT is thrown |  |
| 13 | Call the getValueLength() method | ToolkitException.UNAVAILABLE\_ELEMENT is thrown. |  |
| 14 | Initialize compareBuffer  compareBuffer = 81 11 22 33 44 F5 |  |  |
|  | findAndCompareValue()  tag = 06h, occurrence = 1  valueOffset = 0  compareOffset = 0  compareLength = 6 | Result is 00h |  |
| 15 | Verify current TLV  getValueLength() | Result is 0006 |  |
| 16 | Initialize compareBuffer  compareBuffer = 81 11 22 33 44 F4 |  |  |
|  | Compare buffers with same parameters | Result is +1 |  |
| 17 | Initialize compareBuffer  compareBuffer = 81 11 22 33 44 F6 |  |  |
|  | Compare buffers with same parameters | Result is -1 |  |
| 18 | Initialize compareBuffer  compareBuffer =  55 55 55 22 33 44 F5 55 55 55 55 |  |  |
|  | Compare buffers  findAndCompareValue()  valueOffset = 2  compareOffset = 3  compareLength = 4 | Result is 00h |  |
| 19 | Initialize compareBuffer  compareBuffer =  55 55 55 22 33 45 F5 55 55 55 55 |  |  |
|  | Compare buffers with same parameters | Result is -1 |  |
| 20 | Initialize compareBuffer  compareBuffer =  55 55 55 22 33 43 F5 55 55 55 55 |  |  |
|  | Compare buffers with same parameters | Result is +1 |  |
| 21 | Initialize compareBuffer  compareBuffer =  83 81 55 55 55 55 55 55 55 55 55 55 |  |  |
|  | findAndCompareValue()  tag = 02h, occurrence = 1  valueOffset = 0  compareOffset = 0  compareLength = 2 | Result is 00h |  |
| 22 | Initialize compareBuffer  compareBuffer =  22 44 55 55 55 55 55 55 55 55 55 55 |  |  |
|  | findAndCompareValue()  tag = 02h, occurrence = 2  valueOffset = 0  compareOffset = 0  compareLength = 2 | Result is 00h |  |
| 23 | Initialize compareBuffer  compareBuffer =  22 45 55 55 55 55 55 55 55 55 55 55 |  |  |
|  | findAndCompareValue()  tag = 02h, occurrence = 2  valueOffset = 0  compareOffset = 0  compareLength = 2 | Result is -1 |  |
| 24 | Initialize compareBuffer  compareBuffer =  83 81 55 55 55 55 55 55 55 55 55 55 |  |  |
|  | Successful call (with tag 02h)  findAndCompareValue()  tag = 02h, occurrence = 1  valueOffset = 0  compareBuffer.length = 12  compareOffset = 0  compareLength = 2 | Result is 00h |  |
| 25 | Initialize compareBuffer  compareBuffer = 01 02 … C4 |  |  |
|  | Successful call (with tag B3h)  findAndCompareValue()  tag = B3h, occurrence = 1  valueOffset = 0  compareBuffer.length = 00C4  compareOffset = 0  compareLength = 00C4 | Result is 00h |  |
| 26 | Successful call, findAndCompareValue() with length =0  DstBuffer.length = C4  DstOffset = C4  DstLength = 0 | Result of findAndCompareValue() is 00h |  |

#### 5.2.2.17 Method findAndCopyValue(byte tag, byte[] dstBuffer, short dstOffset)

Test Area Reference: Api\_2\_Ueh\_Facyb\_Bs

5.2.2.17.1 Conformance requirement

The method with following header shall be compliant to its definition in the API.

public short findAndCopyValue(byte tag,

byte[] dstBuffer,

short dstOffset)

throws java.lang.NullPointerException,

java.lang.ArrayIndexOutOfBoundsException,

ToolkitException

5.2.2.17.1.1 Normal execution

- CRRN1: looks for the first occurrence of a TLV element from the beginning of a TLV list and copy its value into a destination buffer.

- CRRN2: if no TLV element is found, the UNAVAILABLE\_ELEMENT exception is thrown and the current TLV is no longer defined.

- CRRN3: if the method is successful then the corresponding TLV becomes current and dstOffset + length of the copied value is returned.

- CRRN4: The search method is comprehension required flag independent.

5.2.2.17.1.2 Parameter errors

- CRRP1: if dstBuffer is null NullPointerException shall be thrown.

- CRRP2: if dstOffset would cause access outside array bounds ArrayIndexOutOfBoundsException shall be thrown.

5.2.2.17.1.3 Context errors

- CRRC1: if the handler is busy an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException.HANDLER\_NOT\_AVAILABLE.

5.2.2.17.2 Test area files

Specific triggering: Unrecognized Envelope

Test Source: Test\_Api\_2\_Ueh\_Facyb\_Bs.java

Test Applet: Api\_2\_Ueh\_Facyb\_Bs\_1.java

Cap File: Api\_2\_Ueh\_Facyb\_Bs.cap

5.2.2.17.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | 9, 11, 13 |
| N2 | 6, 7 |
| N3 | 8, 10, 12 |
| N4 | 14, 15, 16, 17 |
| P1 | 1 |
| P2 | 2, 3, 4, 5 |
| C1 | Does not apply for USATEnvelopeHandler |

5.2.2.17.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
|  | Fill the Unrecognized Envelope with TLV: Tag 02, Value 83 81, Tag 06, Value 81 11 22 33 44 F5, Tag 02 Value 22 44 Tag 33, Length C4 Value 01 02 … |  |  |
| 1 | FindAndCopyValue() with a null dstBuffer | NullPointerException is thrown |  |
| 2 | dstOffset ≥ dstBuffer.length  findAndCopyValue()  tag = 06h  dstBuffer.length = 06  dstOffset = 06 | ArrayIndexOutOfBoundsException is thrown |  |
| 3 | dstOffset < 0  findAndCopyValue()  dstBuffer.length = 06  dstOffset = -1 | ArrayIndexOutOfBoundsException is thrown |  |
| 4 | length > dstBuffer.length  findAndCopyValue()  dstBuffer.length = 05  dstOffset = 0 | ArrayIndexOutOfBoundsException is thrown |  |
| 5 | DstOffset + length >dstBuffer.length  findAndCopyValue()  DstBuffer.length = 06  DstOffset = 1 | ArrayIndexOutOfBoundsException is thrown |  |
| 6 | Select a TLV (tag 02h) |  |  |
|  | findAndCopyValue()  tag = 03h | ToolkitException.UNAVAILABLE\_ELEMENT is thrown |  |
| 7 | Call the getValueLength() method | ToolkitException.UNAVAILABLE\_ELEMENT is thrown. |  |
| 8 | Successful call  findAndCopyValue()  Tag = 06h  DstBuffer.length = 06  DstOffset = 0 | Result of findAndCopyValue () is 0006 |  |
| 9 | Compare buffer  buffer = 81 11 22 33 44 F5 | Result is 00h |  |
| 10 | Initialize dstBuffer  dstBuffer = 55 55 … 55 |  |  |
|  | Successful call  findAndCopyValue()  dstBuffer.length = 12  dstOffset = 2 | Result of findAndCopyValue () is 0008 |  |
| 11 | Compare buffer  buffer =  55 55 81 11 22 33 44 F5 55 55 55 55 | Result is 00h |  |
| 12 | Successful call  findAndCopyValue()  tag = 02h  dstBuffer.length = 2  dstOffset = 0 | Result of findAndCopyValue () is 0002 |  |
| 13 | Compare buffer  buffer = 83 81 | Result is 00h |  |
| 14 | Successful call (with tag 82h)  findAndCopyValue()  tag = 82h  dstBuffer.length = 02  dstOffset = 0 | Result of findAndCopyValue () is 0002 |  |
| 15 | Compare buffer  buffer = 83 81 | Result is 00h |  |
| 16 | Successful call (with tag B3h)  findAndCopyValue()  tag = B3h  dstBuffer.length = C4  dstOffset = 0 | Result of findAndCopyValue () is 00C4 |  |
| 17 | Compare buffer  buffer = 01 02 … C4 | Result is 00h |  |

#### 5.2.2.18 Method findAndCopyValue(byte tag, byte occurrence, short valueOffset, byte[] dstBuffer, short dstOffset, short dstLength)

Test Area Reference: Api\_2\_Ueh\_Facybbs\_Bss

5.2.2.18.1 Conformance requirement

The method with following header shall be compliant to its definition in the API.

public short findAndCopyValue(byte tag,

byte occurrence,

short valueOffset,

byte[] dstBuffer,

short dstOffset,

short dstLength)

throws java.lang.NullPointerException,

java.lang.ArrayIndexOutOfBoundsException,

ToolkitException

5.2.2.18.1.1 Normal execution

- CRRN1: looks for the indicated occurrence of a TLV element from the beginning of a TLV list and copy its value into a destination buffer.

- CRRN2: if no TLV element is found, the UNAVAILABLE\_ELEMENT exception is thrown and the current TLV is no longer defined.

- CRRN3: if the method is successful then the corresponding TLV becomes current and dstOffset + dstLength is returned.

- CRRN4: The search method is comprehension required flag independent.

5.2.2.18.1.2 Parameter errors

- CRRP1: if dstBuffer is null NullPointerException shall be thrown.

- CRRP2: if dstOffset or dstLength or both would cause access outside array bounds, or if dstLength is negative ArrayIndexOutOfBoundsException shall be thrown.

- CRRP3: if valueOffset, dstLength or both are out of the current TLV an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException.OUT\_OF\_TLV\_BOUNDARIES.

5.2.2.18.1.3 Context errors

- CRRC1: if the handler is busy an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException.HANDLER\_NOT\_AVAILABLE.

5.2.2.18.2 Test area files

Specific triggering: Unrecognized Envelope

Test Source: Test\_Api\_2\_Ueh\_Facybbs\_Bss.java

Test Applet: Api\_2\_Ueh\_Facybbs\_Bss\_1.java

Cap File: Api\_2\_Ueh\_Facybbs\_Bss.cap

5.2.2.18.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | 14, 15, 17, 19, 20 |
| N2 | 11, 12 |
| N3 | 13, 15, 17, 19, 25 |
| N4 | 21, 22, 23, 24 |
| P1 | 1 |
| P2 | 2, 3, 4, 5, 6 |
| P3 | 7, 8, 9, 10 |
| C1 | Does not apply for USATEnvelopeHandler |

5.2.2.18.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
|  | Fill the Unrecognized Envelope with TLV: Tag 02, Value 83 81, Tag 06, Value 81 11 22 33 44 F5, Tag 02 Value 22 44 Tag 33, Length C4 Value 01 02 … |  |  |
| 1 | findAndCopyValue() with a null dstBuffer | NullPointerException is thrown |  |
| 2 | dstOffset ≥ dstBuffer.length  findAndCopyValue()  tag = 06h, occurrence = 1  valueOffset = 0  dstBuffer.length = 5  dstOffset = 5  dstLength = 1 | ArrayIndexOutOfBoundsException is thrown |  |
| 3 | dstOffset < 0  findAndCopyValue()  dstBuffer.length = 5  dstOffset = -1  dstLength = 1 | ArrayIndexOutOfBoundsException is thrown |  |
| 4 | dstLength >dstBuffer.length  findAndCopyValue()  dstBuffer.length = 5  dstOffset = 0  dstLength = 6 | ArrayIndexOutOfBoundsException is thrown |  |
| 5 | dstOffset + dstLength >dstBuffer.length  findAndCopyValue()  dstBuffer.length = 5  dstOffset = 3  dstLength = 3 | ArrayIndexOutOfBoundsException is thrown |  |
| 6 | dstLength < 0  findAndCopyValue()  dstBuffer.length = 5  dstOffset = 0  dstLength = -1 | ArrayIndexOutOfBoundsException is thrown |  |
| 7 | valueOffset ≥ Value Length  findAndCopyValue()  tag = 06h, occurrence = 1  valueOffset = 6  dstBuffer.length = 15  dstOffset = 0  dstLength = 1 | ToolkitException.OUT\_OF\_TLV\_BOUNDARIES is thrown |  |
| 8 | valueOffset < 0  findAndCopyValue()  valueOffset = -1  dstBuffer.length = 15  dstOffset = 0  dstLength = 1 | ToolkitException.OUT\_OF\_TLV\_BOUNDARIES is thrown |  |
| 9 | dstLength > Value length  findAndCopyValue()  valueOffset = 0  dstBuffer.length = 15  dstOffset = 0  dstLength = 7 | ToolkitException.OUT\_OF\_TLV\_BOUNDARIES is thrown |  |
| 10 | **valueOffset + dstLength > Text String length**  findAndCopyValue()  valueOffset = 2  dstBuffer.length = 15  dstOffset = 0  dstLength = 5 | ToolkitException.OUT\_OF\_TLV\_BOUNDARIES is thrown |  |
| 11 | Select a TLV (tag 02h) |  |  |
|  | findAndCopyValue()  tag = 06h  occurrence = 2 | ToolkitException.UNAVAILABLE\_ELEMENT is thrown |  |
| 12 | Call the getValueLength() method | ToolkitException.UNAVAILABLE\_ELEMENT is thrown. |  |
| 13 | Successful call  findAndCopyValue()  tag = 06h, occurrence = 1  valueOffset = 0  dstBuffer.length = 06  dstOffset = 0  dstLength = 06 | Result of findAndCopyValue() is 6 |  |
| 14 | Compare buffer  buffer = 81 11 22 33 44 F5 | Result is 00h |  |
| 15 | Initialize dstBuffer  dstBuffer = 55 55 … 55 |  |  |
|  | Successful call  findAndCopyValue()  tag = 06h, occurrence = 1  valueOffset = 2  dstBuffer.length = 12  dstOffset = 3  dstLength = 04 | Result of findAndCopyValue () is 0007 |  |
| 16 | Compare buffer  buffer =  55 55 55 22 33 44 F5 55 55 55 55 55 | Result is 00h |  |
| 17 | Successful call  findAndCopyValue()  tag = 02h, occurrence = 1  valueOffset = 0  dstBuffer.length = 12  dstOffset = 0  dstLength = 2 | Result of findAndCopyValue() is 0002 |  |
| 18 | Compare buffer  buffer = 83 81 55 … 55 | Result is 00h |  |
| 19 | Successful call  findAndCopyValue()  tag = 02h, occurrence = 2  valueOffset = 0  dstBuffer.length = 12  dstOffset = 0  dstLength = 2 | Result of findAndCopyValue() is 0002 |  |
| 20 | Compare buffer  buffer = 22 44 55 … 55 | Result is 00h |  |
| 21 | Successful call (with tag 82h)  findAndCopyValue()  tag = 82h  occurrence = 1  valueOffset = 0  dstBuffer.length = 12  dstOffset = 0  dstLength = 02 | Result of findAndCopyValue () is 0002 |  |
| 22 | Compare buffer  buffer = 83 81 55 … 55 | Result is 00h |  |
| 23 | Successful call (with tag 82h)  findAndCopyValue()  tag = 82h  occurrence = 2  valueOffset = 0  dstBuffer.length = 12  dstOffset = 0  dstLength = 02 | Result of findAndCopyValue () is 0002 |  |
| 24 | Compare buffer  Buffer = 22 44 55 … 55 | Result is 00h |  |
| 25 | Successful call, findAndCopyValue() with length =0  DstBuffer.length = 12  dstOffset = 12  dstLength = 0 | Result of findAndCopyValue () is 12 |  |

#### 5.2.2.19 Method findTLV

Test Area Reference: Api\_2\_Ueh\_Find

5.2.2.19.1 Conformance requirement

The method with following header shall be compliant to its definition in the API.

public byte findTLV(byte tag,

byte occurrence)

throws ToolkitException

5.2.2.19.1.1 Normal execution

Looks for the indicated occurrence of a TLV element from the beginning of the TLV list (handler buffer):

- CRRN1: the method is successful if the required occurrence exists then the corresponding TLV becomes current.

- CRRN2: if the method is successful then it returns TLV\_FOUND\_CR\_SET when Comprehension Required flag is set.

- CRRN3: if the method is successful then it returns TLV\_FOUND\_CR\_NOT\_SET when Comprehension Required flag is not set.

- CRRN4: if the required occurrence of the TLV element does not exist, the current TLV is no longer defined and TLV\_NOT\_FOUND is returned.

- CRRN5: The search method is comprehension required flag independent.

5.2.2.19.1.2 Parameter errors

- CRRP1: if an input parameter is not valid (e.g. occurrence = 0) an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException.BAD\_INPUT\_PARAMETER.

5.2.2.19.1.3 Context errors

- CRRC1: if the handler is busy an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException.HANDLER\_NOT\_AVAILABLE.

5.2.2.19.2 Test area files

Specific triggering: Unrecognized Envelope

Test Source: Test\_Api\_2\_Ueh\_Find.java

Test Applet: Api\_2\_Ueh\_Find\_1.java

Cap File: Api\_2\_Ueh\_Find.cap

5.2.2.19.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | 3, 5 |
| N2 | 2, 4 |
| N3 | 10, 11 |
| N4 | 6, 7, 8, 9 |
| N5 | 12, 13 |
| P1 | 1 |
| C1 | Does not apply for USATEnvelopeHandler |

5.2.2.19.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
|  | Trigger the applet with Unrecognized Envelope including:  Tag 82, tag 86, tag 8B, tag 02 and tag 04 |  |  |
| 1 | Invalid input parameter  findTLV()  Occurrence = 0 | ToolkitException.BAD\_INPUT\_PARAMETER is thrown |  |
| 2 | Search 1st TLV  findTLV()  Tag = 02h  Occurrence = 1 | Result is TLV\_FOUND\_CR\_SET |  |
| 3 | Call the getValueLength() method | Result is 0x02 |  |
| 4 | Search 2nd TLV  findTLV()  Tag = 06h  Occurrence = 1 | Result is TLV\_FOUND\_CR\_SET |  |
| 5 | Call the getValueLength() method | Result is 0x05h |  |
| 6 | Select a TLV (tag 02h) |  |  |
|  | Search a wrong tag  findTLV()  Tag = 03h  Occurrence = 1 | Result is TLV\_NOT\_FOUND |  |
| 7 | Call the getValueLength() method | ToolkitException.UNAVAILABLE\_ELEMENT is thrown. |  |
| 8 | Search a tag with wrong occurrence  findTLV()  Tag = 02h  Occurrence = 3 | Result is TLV\_NOT\_FOUND |  |
| 9 | Call the getValueLength() method | ToolkitException.UNAVAILABLE\_ELEMENT is thrown. |  |
| 10 | Search the TLV  findTLV()  Tag = 02h  Occurrence = 2 | Result is TLV\_FOUND\_CR\_NOT\_SET |  |
| 11 | Search the TLV  findTLV()  Tag = 04h  Occurrence = 1 | Result is TLV\_FOUND\_CR\_NOT\_SET |  |
| 12 | Search tag 86h  findTLV()  Tag = 86h  Occurrence = 1 | Result is TLV\_FOUND\_CR\_SET |  |
| 13 | Search tag 84h  findTLV()  Tag = 84h  Occurrence = 1 | Result is TLV\_FOUND\_CR\_NOT\_SET |  |

#### 5.2.2.20 Method getCapacity

Test Area Reference: Api\_2\_Ueh\_Gcap

5.2.2.20.1 Conformance requirement

The method with following header shall be compliant to its definition in the API.

public byte getCapacity()

5.2.2.20.1.1 Normal execution

- CRRN1: The method shall return the maximum size of the Comprehension TLV list managed by the handler.

5.2.2.20.1.2 Parameter Errors

No requirements

5.2.2.20.1.3 Context errors

No requirements

5.2.2.20.2 Test area files

Test Source: Test\_Api\_2\_Ueh\_Gcap.java

Test Applet: Api\_2\_Ueh\_Gcap\_1.java

Cap File: Api\_2\_Ueh\_Gcap.cap

5.2.2.20.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | 1 |

5.2.2.20.4 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| 1 | **USATEnvelopeHandler available**  1 - Send envelope Menu Selection  2 - The applet calls the getLength() method  3 - The applet calls the getCapacity() method | 1 - Applet is triggered  2 - No exception is thrown  3 - No exception is thrown; the capacity is greater than the BER TLV Length |  |

#### 5.2.2.21 Method getLength

Test Area Reference: Api\_2\_Ueh\_Glen

5.2.2.21.1 Conformance requirement

The method with following header shall be compliant to its definition in the API.

public short getLength()

throws ToolkitException

5.2.2.21.1.1 Normal execution

- CRRN1: returns the length in bytes of the TLV list.

5.2.2.21.1.2 Parameter Errors

No requirements.

5.2.2.21.1.3 Context errors

- CRRC1: if the handler is busy an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException.HANDLER\_NOT\_AVAILABLE.

5.2.2.21.2 Test area files

Specific triggering: Unrecognized envelope

Test Source: Test\_Api\_2\_Ueh\_Glen.java

Test Applet: Api\_2\_Ueh\_Glen\_1.java

Cap File: Api\_2\_Ueh\_Glen.cap

5.2.2.21.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | 1, 2, 3, 4 |
| C1 | Does not apply for USATEnvelopeHandler |

5.2.2.21.4 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| 1 | Send an Unrecognized Envelope with BER length of 0x31 | Result of getLength() is 0x0031 |  |
| 2 | Send an Unrecognized Envelope with BER length of 0x7F | Result of getLength() is 0x007Fh |  |
| 3 | Send an Unrecognized Envelope with BER length of 81 80 | Result of getLength() is 0x0080h |  |
| 4 | Send an Unrecognized Envelope with BER length of 81 FC | Result of getLength() is 0x00FCh |  |

#### 5.2.2.22 Method getValueByte

Test Area Reference: Api\_2\_Ueh\_Gvby

5.2.2.22.1 Conformance requirement

The method with following header shall be compliant to its definition in the API.

public byte getValueByte(short valueOffset)

throws ToolkitException

5.2.2.22.1.1 Normal execution

- CRRN1: Gets a byte from the last TLV element which has been found in the handler and returns its value (1 byte).

5.2.2.22.1.2 Parameter errors

- CRRP1: if valueOffset is out of the current TLV an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException.OUT\_OF\_TLV\_BOUNDARIES.

5.2.2.22.1.3 Context errors

- CRRC1: if the handler is busy an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException.HANDLER\_NOT\_AVAILABLE.

- CRRC2: in case of unavailable TLV element an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException.UNAVAILABLE\_ELEMENT.

5.2.2.22.2 Test area files

Specific triggering: Unrecognized Envelope

Test Source: Test\_Api\_2\_Ueh\_Gvby.java

Test Applet: Api\_2\_Ueh\_Gvby\_1.java

Cap File: Api\_2\_Ueh\_Gvby.cap

5.2.2.22.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | 3, 4, 5, 6, 7, 8 |
| P1 | 2 |
| C1 | Does not apply for USATEnvelopeHandler |
| C2 | 1 |

5.2.2.22.4 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
|  | Fill the Unrecognized envelope with TLV: Tag 02, length 02, value 83 81, Tag 06, length 06, Tag 0B, length 21, Tag 33, Length C8 Value 01 02 … |  |  |
| 1 | getValueByte(0) | ToolkitException.UNAVAILABLE\_ELEMENT is thrown |  |
| 2 | Search TLV 02h |  |  |
|  | getValueByte(2) | ToolkitException.OUT\_OF\_TLV\_BOUNDARIES is thrown |  |
| 3 | Search TLV 02h |  |  |
|  | getValueByte(1) | Result is 0x81 |  |
| 4 | Search TLV 02h (Device Identities TLV) |  |  |
|  | getValueByte(0) | Result is 83h (Source) |  |
| 5 | Search TLV 33h |  |  |
|  | getValueByte(7E) | Result is 0x7F |  |
| 6 | Search TLV 33h |  |  |
|  | getValueByte(80) | Result is 0x81 |  |
| 7 | getValueByte(7F) | Result is 0x80 |  |
| 8 | Search TLV B3h |  |  |
|  | getValueByte(C7) | Result is 0xC8 |  |

#### 5.2.2.23 Method getValueLength

Test Area Reference: Api\_2\_Ueh\_Gvle

5.2.2.23.1 Conformance requirement

The method with following header shall be compliant to its definition in the API.

public short getValueLength()

throws ToolkitException

5.2.2.23.1.1 Normal execution

- CRRN1: gets and returns the binary length of the value field for the last TLV element which has been found in the handler.

5.2.2.23.1.2 Parameter errors

No requirements.

5.2.2.23.1.3 Context errors

- CRRC1: if the handler is busy an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException.HANDLER\_NOT\_AVAILABLE.

- CRRC2: in case of unavailable TLV element an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException.UNAVAILABLE\_ELEMENT.

5.2.2.23.2 Test area files

Specific triggering: Unrecognized Envelope

Test source: Test\_Api\_2\_Ueh\_Gvle.java

Test Applet: Api\_2\_Ueh\_Gvle\_1.java

Cap File: Api\_2\_Ueh\_Gvle.cap

5.2.2.23.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | 2, 3, 4 |
| C1 | Does not apply for USATEnvelopeHandler |
| C2 | 1 |

5.2.2.23.4 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
|  | Fill the Unrecognized envelope with TLV: Tag 02, length 02, Tag 06, length 05, Tag 0B, length 24, Tag 33, Length C8 |  |  |
| 1 | getValueLength() | ToolkitException.UNAVAILABLE\_ELEMENT is thrown |  |
| 2 | Search TLV 02h |  |  |
|  | getValueLength() | Result is 0X0002 |  |
| 3 | Search TLV 0Bh |  |  |
|  | getValueLength() | Result is 0X0024 |  |
| 4 | Search TLV 33h |  |  |
|  | getValueLength() | Result is 0X00C8 |  |

#### 5.2.2.24 Method getValueShort

Test Area Reference: Api\_2\_Ueh\_Gvsh

5.2.2.24.1 Conformance requirement

The method with following header shall be compliant to its definition in the API.

public short getValueShort(short valueOffset)

throws ToolkitException

5.2.2.24.1.1 Normal execution

- CRRN1: Gets a short from the last TLV element which has been found in the handler and returns its value (1 short).

5.2.2.24.1.2 Parameter errors

- CRRP1: if valueOffset is out of the current TLV an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException.OUT\_OF\_TLV\_BOUNDARIES.

5.2.2.24.1.3 Context errors

- CRRC1: if the handler is busy an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException.HANDLER\_NOT\_AVAILABLE.

- CRRC2: in case of unavailable TLV element an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException.UNAVAILABLE\_ELEMENT.

5.2.2.24.2 Test area files

Specific triggering: Unrecognized Envelope

Test Source: Test\_Api\_2\_Ueh\_Gvsh.java

Test Applet: Api\_2\_Ueh\_Gvsh\_1.java

Cap File: Api\_2\_Ueh\_Gvsh.cap

5.2.2.24.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | 3, 4, 5, 6, 7, 8 |
| P1 | 2 |
| C1 | Does not apply for USATEnvelopeHandler |
| C2 | 1 |

5.2.2.24.4 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
|  | Fill the Unrecognized envelope with TLVs:  Tag 02, Length 02 Value 83 81  Tag 06, Length 06 Value 81 11 22 33 44 F5  Tag 33, Length C9 Value 01 02 … |  |  |
| 1 | getValueShort(0) | ToolkitException.UNAVAILABLE\_ELEMENT is thrown |  |
| 2 | Search TLV 02h |  |  |
|  | getValueShort(2) | ToolkitException.OUT\_OF\_TLV\_BOUNDARIES is thrown |  |
| 3 | Search TLV 02h |  |  |
|  | getValueShort(0) | Result is 0x83 0x81 |  |
| 4 | Search TLV 06h |  |  |
|  | getValueShort(1) | Result is 0x11 0x22 |  |
| 5 | Search TLV 33h |  |  |
|  | getValueShort(7E) | Result is 0x7F 0x80 |  |
| 6 | Search TLV 33h |  |  |
|  | getValueShort(80) | Result is 0x81 0x82 |  |
| 7 | getValueShort(7F) | Result is 0x80 0x81 |  |
| 8 | Search TLV B3h |  |  |
|  | getValueShort(C7) | Result is 0xC8 0xC9 |  |

### 5.2.3 Interface USATTerminalProfile

The constants in Java are resolved at compilation time, therefore a runtime test is not useful. No test of constants will be performed

### 5.2.4 Class USATEnvelopeHandlerSystem

#### 5.2.4.1 Method getTheHandler

Test Area Reference: Api\_2\_Ues\_Gthd

5.2.4.1.1 Conformance requirements

The method with following header shall be compliant to its definition in the API.

public static USATEnvelopeHandler getTheHandler()

throws uicc.toolkit.ToolkitException

5.2.4.1.1.1 Normal execution

- CRRN1: The method shall return the single system instance of the class implementing the USATEnvelopeHandler interface.

- CRRN2: The USATEnvelopeHandler is a Temporary JCRE Entry Point Object (see Java Card 2.2.1 Runtime Environment (JCRE) Specification [])

5.2.4.1.1.2 Parameter Errors

No requirements.

5.2.4.1.1.3 Context errors

- CRRC1: The method shall throw ToolkitException.HANDLER\_NOT\_AVAILABLE if the handler is busy.

5.2.4.1.2 Test area files

Test Source: Test\_Api\_2\_Ues\_Gthd.java

Test Applet: Api\_2\_Ues\_Gthd\_1.java

Cap File: Api\_2\_Ues\_Gthd.cap

5.2.4.1.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | 1, 2, 3 |
| N2 | Tested in clause 5.3.7.1, other parts transferred to (U)SAT framework from API. |
| C1 | Tested in clause 5.3.1.5, Minimum Handler Availability on USATEnvelopeHandler. |

5.2.4.1.4 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| 1 | Call GetTheHandler() method twice | The returned objects shall be the same |  |
| 2 | Verify that getTheHandler() method returns an USATEnvelopeHandler | The reference returned shall be an object implementing the USATEnvelopeHandler interface (check cast) |  |
| 3 | Verify the returned value is not null | The reference returned shall not be null. |  |

### 5.2.5 Interface ToolkitRegistry

#### 5.2.5.1 Method clearEvent

Test Area Reference: Api\_2\_Tkr\_Cevt

5.2.5.1.1 Conformance requirement:

The method with following header shall be compliant to its definition in the API.

public void clearEvent(short event)

throws ToolkitException,

javacard.framework.TransactionException

5.2.5.1.1.1 Normal execution

- CRRN1: A call to isEventSet() method for a cleared event should return false after a call to clearEvent.

- CRRN2: The (U)SAT Framework shall not trigger the applet on the occurrence of the cleared event anymore.

- CRRN3: if event was EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM and after the call, no applet is registered to it, The (U)SAT Framework shall allow an applet to register to this event.

- CRRN4: if event was EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM and one applet is still registered to this event, The (U)SAT Framework shall not allow an applet to register to this event.

5.2.5.1.1.2 Parameter Errors

No requirements.

5.2.5.1.1.3 Context errors

- CRRC1: shall throw javacard.framework.TransactionException - if the operation would cause the commit capacity to be exceeded.

5.2.5.1.2 Test area files

Test Source: Test\_Api\_2\_Tkr\_Cevt.java

Test Applet: Api\_2\_Tkr\_Cevt\_1.java

Cap File: Api\_2\_Tkr\_Cevt.cap

5.2.5.1.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | 1 |
| N2 | 2 |
| N3 | Framework |
| N4 | Framework |
| C1 | not testable |

5.2.5.1.4 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| 1 | Clear ALLOWED events  Install Applet registered to EVENT\_FORMATTED\_SMS\_PP\_ENV event  For events (2 to 6, 10 and 24) defined in TS 31.130 [2]:  EVENT\_FORMATTED\_SMS\_PP\_ENV  EVENT\_FORMATTED\_SMS\_PP\_UPD  EVENT\_UNFORMATTED\_SMS\_PP\_ENV  EVENT\_UNFORMATTED\_SMS\_PP\_UPD  EVENT\_UNFORMATTED\_SMS\_CB  EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM  EVENT\_FORMATTED\_SMS\_CB  EVENT\_FORMATTED\_USSD  EVENT\_UNFORMATTED\_USSD  EVENT\_DOWNLOAD\_IWLAN\_ACCESS\_STATUS  The applet calls:  1- Call clearEvent() method  2- Call isEventSet() method | 1- No exception is thrown each time.  2- Shall return false each time. |  |
| 2 | Checking applet isn't triggered by an ENVELOPE(EVENT\_FORMATTED\_SMS\_PP\_ENV) command  1 - reset and initialize the card  2 - An ENVELOPE(EVENT\_FORMATTED\_SMS\_PP\_ENV) is sent. | Applet is not triggered by an ENVELOPE(EVENT\_FORMATTED\_SMS\_PP\_ENV) command |  |

#### 5.2.5.2 Method isEventSet

Test Area Reference: Api\_2\_Tkr\_Ievs

5.2.5.2.1 Conformance requirement:

The method with following header shall be compliant to its definition in the API.

public boolean isEventSet(short event)

5.2.5.2.1.1 Normal execution

- CRRN1: shall return true if the event is set in the Toolkit Registry for the applet.

- CRRN2: shall return false if the event is not set in the Toolkit Registry for the applet.

5.2.5.2.1.2 Parameter errors

No requirements.

5.2.5.2.1.3 Context errors

No requirements.

5.2.5.2.2 Test area files

Test Source: Test\_Api\_2\_Tkr\_Ievs.java

Test Applet: Api\_2\_Tkr\_Ievs\_1.java

Cap File: Api\_2\_Tkr\_Ievs.cap

5.2.5.2.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | 2 |
| N2 | 1 |

5.2.5.2.4 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| 1 | Install Applet only registered to  EVENT\_EVENT\_DOWNLOAD\_USER\_ACTIVITY  Test that events are not set  Applet calls isEventSet() method for each event ranging from (2 to 6, 10 and 24) | Shall return false each time. |  |
| 2 | Setting events  For the following events defined in TS 31.130 [2] for setEvent() method:  EVENT\_FORMATTED\_SMS\_PP\_ENV  EVENT\_FORMATTED\_SMS\_PP\_UPD  EVENT\_UNFORMATTED\_SMS\_PP\_ENV  EVENT\_UNFORMATTED\_SMS\_PP\_UPD  EVENT\_UNFORMATTED\_SMS\_CB  EVENT\_FORMATTED\_USSD  EVENT\_UNFORMATTED\_USSD  EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM  EVENT\_FORMATTED\_SMS\_CB  EVENT\_DOWNLOAD\_IWLAN\_ACCESS\_STATUS  applet calls:  1- Call setEvent() method  2- Call isEventSet() method | 1- No exception shall be thrown.  2- Shall return true each time. |  |

#### 5.2.5.3 Method setEvent

Test Area Reference: Api\_2\_Tkr\_Sevt

5.2.5.3.1 Conformance requirement:

The method with following header shall be compliant to its definition in the API.

public void setEvent(short id)

throws ToolkitException,

javacard.framework.TransactionException

5.2.5.3.1.1 Normal execution

- CRRN1: a following call to isEventSet() method with the same event id shall answer true for the applet.

- CRRN2: the (U)SAT Framework shall trigger the applet if an occurrence of the set event happens.

5.2.5.3.1.2 Parameter errors

No requirements.

5.2.5.3.1.3 Context errors

- CRRC1: shall throw a ToolkitException with EVENT\_ALREADY\_REGISTERED if event is EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM but another applet is already registered to it.

- CRRC2: shall throw a ToolkitException with EVENT\_ALREADY\_REGISTERED if event is EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM but another applet that it is not in selectable state is already registered to it.

- CRRC3: shall throw a ToolkitException with TAR\_NOT\_DEFINED if event is FORMATTED\_SMS\_PP\_ENV and the applet has no TAR defined.

- CRRC4: shall throw a ToolkitException with TAR\_NOT\_DEFINED if event is FORMATTED\_SMS\_PP\_UPD and the applet has no TAR defined.

- CRRC5: shall throw a ToolkitException with TAR\_NOT\_DEFINED if event is FORMATTED\_SMS\_CB and the applet has no TAR defined.

- CRRC6: shall throw a ToolkitExceptin TAR\_NOT\_DEFINED if event is FORMATTED\_USSD and the applet has no TAR defined.

5.2.5.3.2 Test area files

Test Source: Test\_Api\_2\_Tkr\_Sevt.java

Test Applet: Api\_2\_Tkr\_Sevt\_1.java

Api\_2\_Tkr\_Sevt\_2.java

Api\_2\_Tkr\_Sevt\_3.java

The load script installs the 3 instances.

Cap File: Api\_2\_Tkr\_Sevt.cap

5.2.5.3.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | 1 |
| N2 | 2, 3 |
| C1 | 4 |
| C2 | 5 |
| C3 | 6 |
| C4 | 6 |
| C5 | 6 |
| C6 | 6 |

5.2.5.3.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
| 1 | Setting events  1- For events (2 to 6, 10 and 24) defined in TS 31.130 [2]:  EVENT\_FORMATTED\_SMS\_PP\_ENV  EVENT\_FORMATTED\_SMS\_PP\_UPD  EVENT\_UNFORMATTED\_SMS\_PP\_ENV  EVENT\_UNFORMATTED\_SMS\_PP\_UPD  EVENT\_UNFORMATTED\_SMS\_CB  EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM  EVENT\_FORMATTED\_SMS\_CB  EVENT\_FORMATTED\_USSD  EVENT\_UNFORMATTED\_USSD  EVENT\_IWLAN\_ACCESS\_STATUS  1.1- Call clearEvent(event)  1.2- Call isEventSet(event)  1.3- Call setEvent(event)  1.4- Call isEventSet(event)  1.5- Call clearEvent(event) | 1.1- No exception shall be thrown.  1.2- Shall return false.  1.3- No exception shall be thrown.  1.4- Shall return true.  1.5- No exception shall be thrown. |  |
| 2 | Setting EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM  Call setEvent(EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM) | No Exception shall be thrown |  |
| 3 | Check applet is triggered by an ENVELOPE (EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM)  Trigger the applet | Applet is triggered by an ENVELOPE (EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM) |  |
| 4 | Applet2 registers to EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM but it is already assigned  1- Trigger Applet2 by ENVELOPE(EVENT\_FORMATTED\_SMS\_PP\_ENV)  2- Applet2 call setEvent(EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM) | 2- Shall throw a ToolkitException with EVENT\_ALREADY\_REGISTERED reason code. |  |
| 5 | Applet2 registers to EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM but it is already assigned to another applet in not selectable state  1- Set the Applet1 in the lock state  2- Applet2 call setEvent (EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM)  3- Set the Applet1 in the make selectable state | 2- Shall throw a ToolkitException with EVENT\_ALREADY\_REGISTERED reason code. |  |
| 6 | Applet3 with no TAR defined registers to EVENT\_UNFORMATTED\_SMS\_CB event  1- send unformatted ENVELOPE(CELL\_BROADCAST\_DATA\_DOWNLOAD)  2- setEvent(EVENT\_FORMATTED\_SMS\_PP\_ENV)  3- setEvent(EVENT\_FORMATTED\_SMS\_PP\_UPD)  4- setEvent(EVENT\_FORMATTED\_SMS\_CB)  5- setEvent(EVENT\_FORMATTED\_USSD) | 1- Applet3 shall be triggered  2- ToolkitException with reason code TAR\_NOT\_DEFINED shall be thrown  3- ToolkitException with reason code TAR\_NOT\_DEFINED shall be thrown  4- ToolkitException with reason code TAR\_NOT\_DEFINED shall be thrown  5- ToolkitException with reason code  TAR\_NOT\_DEFINED shall be thrown |  |

#### 5.2.5.4 Method setEventList

Test Area Reference: Api\_2\_Tkr\_Sevl

5.2.5.4.1 Conformance requirement:

The method with following header shall be compliant to its definition in the API.

public void setEventList(short[] eventList,

short offset,

short length)

throws java.lang.NullPointerException,

java.lang.ArrayIndexOutOfBoundsException,

ToolkitException,

javacard.framework.TransactionException

5.2.5.4.1.1 Normal execution

- CRRN1: for all events set successfully by this method, a call to isEventSet() method should return true.

- CRRN2: the (U)SAT Framework shall trigger the applet if an occurrence of one of the successfully registered events happens.

5.2.5.4.1.2 Parameter errors

No requirements.

5.2.5.4.1.3 Context errors

- CRRC1: shall throw a ToolkitException with EVENT\_ALREADY\_REGISTERED if eventList contains EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM but another applet is already registered to it.

- CRRC2: shall throw a ToolkitException with EVENT\_ALREADY\_REGISTERED if event is EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM but another applet that it is not in selectable state is already registered to it.

- CRRC3: shall throw a ToolkitException with TAR\_NOT\_DEFINED if event is FORMATTED\_SMS\_PP\_ENV and the applet has no TAR defined.

- CRRC4: shall throw a ToolkitException with TAR\_NOT\_DEFINED if event is FORMATTED\_SMS\_PP\_UPD and the applet has no TAR defined.

- CRRC5: shall throw a ToolkitException with TAR\_NOT\_DEFINED if event is FORMATTED\_SMS\_CB and the applet has no TAR defined.

- CRRC6: shall throw a ToolkitException with TAR\_NOT\_DEFINED if event is FORMATTED\_USSD and the applet has no TAR defined

5.2.5.4.2 Test area files

Test Source: Test\_Api\_2\_Tkr\_Sevl.java

Test Applet: Api\_2\_Tkr\_Sevl\_1.java

Api\_2\_Tkr\_Sevl\_2.java

Cap File: Api\_2\_Tkr\_Sevl.cap

5.2.5.4.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | 1,2 |
| N2 | 3, 4 |
| C1 | 5 |
| C2 | 6 |
| C3 | 7 |
| C4 | 7 |
| C5 | 7 |
| C6 | 7 |

5.2.5.4.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
| 1 | Applet1 registers all eventList buffer  EventList = events (2 to 6, 10 and 24) defined in TS 31.130 [2]:  EVENT\_FORMATTED\_SMS\_PP\_ENV  EVENT\_FORMATTED\_SMS\_PP\_UPD  EVENT\_UNFORMATTED\_SMS\_PP\_ENV  EVENT\_UNFORMATTED\_SMS\_PP\_UPD  EVENT\_UNFORMATTED\_SMS\_CB  EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM  EVENT\_FORMATTED\_SMS\_CB  EVENT\_FORMATTED\_USSD  EVENT\_UNFORMATTED\_USSD  EVENT\_DOWNLOAD\_IWLAN\_ACCESS\_STATUS  1- For each event in EventList clearEvent(event)  2- Call setEventList(eventList)  Offset = 0  Length = eventList.length  3- For all events in eventList isEventSet(event)  4- For each event in EventList clearEvent(event) | 1- No exception shall be thrown.  2- No exception shall be thrown.  3- Each time shall return true.  4- No exception shall be thrown. |  |
| 2 | Registering part of eventList buffer  EventList = events (2 to 6, 10 and 24) defined in TS 31.130 [2]:  EVENT\_FORMATTED\_SMS\_PP\_ENV  EVENT\_FORMATTED\_SMS\_PP\_UPD  EVENT\_UNFORMATTED\_SMS\_PP\_ENV  EVENT\_UNFORMATTED\_SMS\_PP\_UPD  EVENT\_UNFORMATTED\_SMS\_CB  EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM  EVENT\_FORMATTED\_SMS\_CB  EVENT\_FORMATTED\_USSD  EVENT\_UNFORMATTED\_USSD  EVENT\_DOWNLOAD\_IWLAN\_ACCESS\_STATUS  1- For each event in EventList clearEvent(event)  2- setEventList(eventList, offset, length)  Offset > 0  Length = eventList.lentgh – offset  3- For all events in eventList:  Call isEventSet(event)  4- For each event in EventList: clearEvent(event) | 1- No exception shall be thrown.  2- No exception shall be thrown.  3- Each time shall return true for events ranging from offset to offset+length else shall return false.  4- No exception shall be thrown. |  |
| 3 | Setting EVENT\_MO\_SHORT\_MESSAGE\_ CONTROL\_BY\_SIM  Call setEventList(MonoEventList, 0, 1) with MonoEventList containing  EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM | Shall not throw an exception |  |
| 4 | Check applet1 is triggered by an ENVELOPE (EVENT\_MO\_SHORT\_MESSAGE\_ CONTROL\_BY\_SIM)  Reset and initialize the card  Trigger the applet | Applet is triggered by an ENVELOPE (EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM) |  |
| 5 | Applet 2 registers to EVENT\_MO\_SHORT\_MESSAGE\_ CONTROL\_BY\_SIM but it is already assigned  Call setEventList(MonoEventList,0,1) with MonoEventList containing EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM | Shall throw a ToolkitException with EVENT\_ALREADY\_REGISTERED reason code. |  |
| 6 | Applet2 registers to EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM but it is already assigned to another applet in not selectable state  1- Set the Applet1 in the lock state  2- Applet2 calls setEventList(MonoEventList,0,1) with MonoEventList containing EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM  3- Set the Applet1 in the make selectable state | 2- Shall throw a ToolkitException with EVENT\_ALREADY\_REGISTERED reason code. |  |
| 7 | Applet3 with no TAR defined registers to EVENT\_UNFORMATTED\_SMS\_CB event  1- send unformatted ENVELOPE(CELL\_BROADCAST\_DATA\_DOWNLOAD)  2- setEventList(EventList,0,1) with EventList containing EVENT\_FORMATTED\_SMS\_PP\_ENV  3- setEventList (EventList,1,1) with EventList containing EVENT\_FORMATTED\_SMS\_PP\_UPD  4- setEventList (EventList,2,1) with EventList containing EVENT\_FORMATTED\_SMS\_CB  5- isEventSet(EVENT\_FORMATTED\_SMS\_PP\_ENV)  6- isEventSet(EVENT\_FORMATTED\_SMS\_PP\_UPD)  7- isEventSet(EVENT\_FORMATTED\_SMS\_CB)  8- isEventSet(EVENT\_FORMATTED\_USSD) | 1- Applet3 shall be triggered  2- ToolkitException with reason code TAR\_NOT\_DEFINED shall be thrown  3- ToolkitException with reason code TAR\_NOT\_DEFINED shall be thrown  4- ToolkitException with reason code TAR\_NOT\_DEFINED shall be thrown  5- method shall return FALSE  6- method shall return FALSE  7- method shall return FALSE  8- method shall return FALSE |  |

## 5.3 (U)SAT Framework

### 5.3.1 Minimum handler availability

#### 5.3.1.1 ProactiveHandler

Test Area Reference: Ufw\_Mha\_Pahd

5.3.1.1.1 Conformance requirements

5.3.1.1.1.1 Normal execution

- CRRN1: If a proactive session is not ongoing the *ProactiveHandler* is available from the invocation to the termination of the *processToolkit()* method for the following events:

EVENT\_FORMATTED\_SMS\_PP\_ENV

EVENT\_FORMATTED\_SMS\_PP\_UPD

EVENT\_UNFORMATTED\_SMS\_PP\_ENV

EVENT\_UNFORMATTED\_SMS\_PP\_UPD

EVENT\_UNFORMATTED\_SMS\_CB

EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM

EVENT\_FORMATTED\_SMS\_CB

EVENT\_FORMATTED\_USSD

EVENT\_UNFORMATTED\_USSD

EVENT\_DOWNLOAD\_IWLAN\_ACCESS\_STATUS

- CRRN2: A *ProactiveHandler* is considered available when no HANDLER\_NOT\_AVAILABLE *ToolkitException* is thrown when the corresponding *getTheHandler()* method is called or a method of the handler is called.

- CRRN3: When available the *ProactiveHandler* shall remain available until the termination of the *processToolkit()* method.

- CRRN4: If a proactive command is pending the *ProactiveHandler* may not be available.

5.3.1.1.1.2 Parameter errors

No requirements.

5.3.1.1.1.3 Context errors

- CRRC1: The *ProactiveHandler* shall not be available if the Terminal Profile command has not yet been processed by the (U)SAT Framework.

5.3.1.1.2 Test area files

Test Source: Test\_Ufw\_Mha\_Pahd.java

Test Applet: Ufw\_Mha\_Pahd\_1.java

Ufw\_Mha\_Pahd\_2.java

Cap File: Ufw\_Mha\_Pahd.cap

5.3.1.1.3 Test coverage

| CRR Number | Test Case Number |
| --- | --- |
| N1 | 1 to 10 |
| N2 | 11 to 20 |
| N3 | 1 to 11 |
| N4 | Not testable |
| C1 | 12 to 17 and also tested in Test Cases 8 to 14 in Ufw\_Mha\_Prhd. Applicable only if the applet is triggerred when no terminal profile has been previously received. |

5.3.1.1.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
| 1 | ProactiveHandler availability with EVENT\_FORMATTED\_SMS\_PP\_ENV  1- Envelope SMS-PP Download formatted is sent to the (U)SIM  2- Applet1 gets the ProactiveHandler | 1- Applet1 is triggered  2- No exception is thrown.  Applet1 finalizes |  |
| 2 | ProactiveHandler availability with EVENT\_FORMATTED\_SMS\_PP\_UPD  1- Update Record EFSMS instruction formatted is sent to the (U)SIM  2- Applet1 gets the ProactiveHandler | 1- Applet1 is triggered  2- No exception is thrown.  Applet1 finalizes |  |
| 3 | ProactiveHandler availability with EVENT\_UNFORMATTED\_SMS\_PP\_ENV  1- Envelope SMS-PP Download unformatted is sent to the (U)SIM  2- Applet1 gets the ProactiveHandler  3- Applet2 gets the ProactiveHandler | 1- Applet1 is triggered  2- No exception is thrown.  Applet1 finalizes  Applet2 is triggered  3- No exception is thrown. |  |
| 4 | ProactiveHandler availability with EVENT\_UNFORMATTED\_SMS\_PP\_UPD  1- Update Record EFSMS instruction unformatted is sent to the (U)SIM  2- Applet1 gets the ProactiveHandler  3- Applet2 gets the ProactiveHandler | 1- Applet1 is triggered  2- No exception is thrown.  Applet1 finalizes  3- Applet2 is triggered  4- No exception is thrown. |  |
| 5 | ProactiveHandler availability with EVENT\_UNFORMATTED\_SMS\_CB  1- Envelope Cell Broadcast Download unformatted is sent to the (U)SIM  2- Applet1 gets the ProactiveHandler  3- Applet2 gets the ProactiveHandler | 1- Applet1 is triggered  2- No exception is thrown  Applet1 finalizes  Applet2 is triggered  3- No exception is thrown |  |
| 6 | ProactiveHandler availability with EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM  1- Envelope MO short message control by SIM is sent to the (U)SIM  2- Applet1 gets the ProactiveHandler | 1- Applet1 is triggered  2- No exception is thrown |  |
| 7 | ProactiveHandler availability with EVENT\_FORMATTED\_SMS\_CB  1- Envelope Cell Broadcast Download formatted is sent to the (U)SIM  2- Applet1 gets the ProactiveHandler | 1- Applet1 is triggered  2-No exception is thrown  Applet1 finalizes |  |
| 8 | ProactiveHandler availability with EVENT\_FORMATTED\_USSD  1- Envelope USSD formatted is sent to the (U)SIM  2- Applet1 gets the ProactiveHandler | 1- Applet1 is triggered  2- No exception is thrown.  Applet1 finalizes |  |
| 9 | ProactiveHandler availability with EVENT\_UNFORMATTED\_USSD  1- Envelope USSD unformatted is sent to the (U)SIM  2- Applet1 gets the ProactiveHandler  3- Applet2 gets the ProactiveHandler | 1- Applet1 is triggered  2- No exception is thrown.  Applet1 finalizes  Applet2 is triggered  3- No exception is thrown. |  |
| 10 | ProactiveHandler availability with EVENT\_DOWNLOAD\_IWLAN\_ACCESS\_STATUS  1- Envelope DOWNLOAD\_IWLAN\_ACCESS\_STATUS is sent to the (U)SIM  2- Applet1 gets the ProactiveHandler  **3- Applet2 gets** the **ProactiveHandler** | Applet1 is triggered  2- No exception is thrown.  Applet1 finalizes  Applet2 is triggered  3- No exception is thrown. |  |
| 11 | The ProactiveHandler is not available before the Terminal Profile with EVENT\_FORMATTED\_SMS\_PP\_ENV  1- Reset the card without sending the Terminal Profile  2- Envelope SMS-PP Download formatted is sent to the (U)SIM  3- Applet1 gets the ProactiveHandler | 2- Applet1 is triggered  3- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet1 finalizes |  |
| 12 | The ProactiveHandler is not available before the Terminal Profile with EVENT\_FORMATTED\_SMS\_PP\_UPD  1- Update Record EFSMS instruction formatted is sent to the (U)SIM  2- Applet1 gets the ProactiveHandler | 1- Applet1 is triggered  2- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet1 finalizes |  |
| 13 | The ProactiveHandler is not available before the Terminal Profile with EVENT\_UNFORMATTED\_SMS\_PP\_ENV  1- Envelope SMS-PP Download unformatted is sent to the (U)SIM  2- Applet1 gets the ProactiveHandler  3- Applet2 gets the ProactiveHandler | 1- Applet1 is triggered  2- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet1 finalizes  Applet2 is triggered  3- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet2 finalizes |  |
| 14 | The ProactiveHandler is not available before the Terminal Profile with EVENT\_UNFORMATTED\_SMS\_PP\_UPD  1- Update Record EFSMS instruction unformatted is sent to the (U)SIM  2- Applet1 gets the ProactiveHandler  3- Applet2 gets the ProactiveHandler | 1- Applet1 is triggered  2- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet1 finalizes  Applet2 is triggered  3- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet2 finalizes |  |
| 15 | The ProactiveHandler is not available before the Terminal Profile with EVENT\_UNFORMATTED\_SMS\_CB  1- Envelope Cell Broadcast Download unformatted is sent to the (U)SIM  2- Applet1 gets the ProactiveHandler  3- Applet2 gets the ProactiveHandler | 1- Applet1 is triggered  2- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet1 finalizes  Applet2 is triggered  3- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet2 finalizes |  |
| 16 | The ProactiveHandler is not available before the Terminal Profile with EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM  1- Envelope MO short message control by SIM is sent to the (U)SIM  2- Applet1 gets the ProactiveHandler | 1- Applet1 is triggered  2- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet1 finalizes |  |
| 17 | The ProactiveHandler is not available before the Terminal Profile with EVENT\_FORMATTED\_SMS\_CB  1- Envelope Cell Broadcast Download formatted is sent to the (U)SIM  2- Applet1 gets the ProactiveHandler | 1- Applet1 is triggered  2- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet1 finalizes |  |
| 18 | The ProactiveHandler is not available before the Terminal Profile with EVENT\_FORMATTED\_USSD  1- Envelpe USSD formatted is sent to the (U)SIM  2- Applet gets the ProactiveHandler | 1- Applet1 is triggered  2- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet1 finalizes |  |
| 19 | The ProactiveHandler is not available before the Terminal Profile with EVENT\_UNFORMATTED\_USSD  1- Envelope USSD unformatted is sent to the (U)SIM  2- Applet1 gets the ProactiveHandler  3- Applet2 gets the ProactiveHandler | 1- Applet1 is triggered  2- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet1 finalizes  Applet2 is triggered  3- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet2 finalizes |  |
| 20 | The ProactiveHandler is not available befor the Terminal Profile with EVENT\_DOWNLOAD\_IWLNA\_ACCESS\_STATUS  1- Envelope Download Iwlan Access Status h n is sent to the (U)SIM  2- Applet1 gets the ProactiveHandler  3- Applet2 gets the ProactiveHandler | 1- Applet1 is triggered  2- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet1 finalizes  Applet2 is triggered  3- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet2 finalizes |  |

#### 5.3.1.2 ProactiveResponseHandler

Test Area Reference: Ufw\_Mha\_Prhd

5.3.1.2.1 Conformance requirements

5.3.1.2.1.1 Normal execution

- CRRN1: The *ProactiveResponseHandler* is available as soon as the *ProactiveHandler* is available and remains available until the termination of the *processToolkit()* method for the following events:

EVENT\_FORMATTED\_SMS\_PP\_ENV

EVENT\_FORMATTED\_SMS\_PP\_UPD

EVENT\_UNFORMATTED\_SMS\_PP\_ENV

EVENT\_UNFORMATTED\_SMS\_PP\_UPD

EVENT\_UNFORMATTED\_SMS\_CB

EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM

EVENT\_FORMATTED\_SMS\_CB

EVENT\_FORMATTED\_USSD

EVENT\_UNFORMATTED\_USSD

EVENT\_DOWNLOAD\_IWLAN\_ACCESS\_STATUS

- CRRN2: A *ProactiveResponseHandler* is considered available when no HANDLER\_NOT\_AVAILABLE ToolkitException is thrown when the corresponding *getTheHandler()* method is called or a method of the handler is called.

5.3.1.2.1.2 Parameter errors

No requirements.

5.3.1.2.1.3 Context errors

- CRRC1: The *ProactiveResponseHandler* shall not be available if the *ProactiveHandler* is not available.

5.3.1.2.2 Test area files

Test Source: Test\_Ufw\_Mha\_Prhd.java

Test Applet: Ufw\_Mha\_Prhd\_1.java

Ufw\_Mha\_Prhd\_2.java

Cap File: Ufw\_Mha\_Prhd.cap

5.3.1.2.3 Test coverage

| CRR Number | Test Case Number |
| --- | --- |
| N1 | 1 to 10 |
| N2 | 1 to 20 |
| C1 | 8 to 20.  Applicable only if the applet is triggerred when no terminal profile has been previously received. |

5.3.1.2.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
| 1 | ProactiveResponseHandler availability with EVENT\_FORMATTED\_SMS\_PP\_ENV  1- Envelope SMS-PP Download formatted is sent to the (U)SIM  Applet builds a proactive command DISPLAY TEXT  2- ProactiveHandler.send() method is called  3- ProactiveResponseHandler.getTheHandler() method is called | 1- Applet1 is triggered  3- No exception is thrown | 2- A proactive command DISPLAY TEXT is fetched  TERMINAL RESPONSE |
| 2 | ProactiveResponseHandler availability with EVENT\_FORMATTED\_SMS\_PP\_UPD  1- Update Record EFSMS instruction formatted is sent to the (U)SIM  Applet builds a proactive command DISPLAY TEXT  2- ProactiveHandler.send() method is called  3- ProactiveResponseHandler.getTheHandler() method is called | 1- Applet1 is triggered  3- No exception is thrown | 2- A proactive command DISPLAY TEXT is fetched  TERMINAL RESPONSE |
| 3 | ProactiveResponseHandler availability with EVENT\_UNFORMATTED\_SMS\_PP\_ENV  1- Envelope SMS-PP Download unformatted is sent to the (U)SIM  Applet1 builds a proactive command DISPLAY TEXT  2- ProactiveHandler.send() method is called  3- ProactiveResponseHandler.getTheHandler() method is called  Applet2 builds a proactive command DISPLAY TEXT  4- ProactiveHandler.send() method is called  5- ProactiveResponseHandler.getTheHandler() method is called | 1- Applet1 is triggered  3- No exception is thrown  Applet1 finalizes  Applet2 is triggered  5- No exception is thrown | 2- A proactive command DISPLAY TEXT is fetched  TERMINAL RESPONSE  4- A proactive command DISPLAY TEXT is fetched  TERMINAL RESPONSE |
| 4 | ProactiveResponseHandler availability with EVENT\_UNFORMATTED\_SMS\_PP\_UPD  1- Update Record EFSMS instruction unformatted is sent to the (U)SIM  Applet1 builds a proactive command DISPLAY TEXT  2- ProactiveHandler.send() method is called  3- ProactiveResponseHandler.getTheHandler() method is called  Applet2 builds a proactive command DISPLAY TEXT  4- ProactiveHandler.send() method is called  5- ProactiveResponseHandler.getTheHandler() method is called | 1- Applet1 is triggered  3- No exception is thrown  Applet1 finalizes  Applet2 is triggered  5- No exception is thrown | 2- A proactive command DISPLAY TEXT is fetched  TERMINAL RESPONSE  4- A proactive command DISPLAY TEXT is fetched  TERMINAL RESPONSE |
| 5 | ProactiveResponseHandler availability with EVENT\_UNFORMATTED\_SMS\_CB  1- Envelope Cell Broadcast Download unformatted is sent to the (U)SIM  Applet1 builds a proactive command DISPLAY TEXT  2- ProactiveHandler.send() method is called  3- ProactiveResponseHandler.getTheHandler() method is called.  Applet2 builds a proactive command DISPLAY TEXT  4- ProactiveHandler.send() method is called  5- ProactiveResponseHandler.getTheHandler() method is called | 1- Applet1 is triggered  3- No exception is thrown  Applet1 finalizes  Applet2 is triggered  5- No exception is thrown | 2- A proactive command DISPLAY TEXT is fetched  TERMINAL RESPONSE  4- A proactive command DISPLAY TEXT is fetched  TERMINAL RESPONSE |
| 6 | ProactiveResponseHandler availability with EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM  1- Envelope MO short message control by SIM is sent to the (U)SIM  Applet builds a proactive command DISPLAY TEXT  2- ProactiveHandler.send() method is called  3- ProactiveResponseHandler.getTheHandler() method is called | 1- Applet1 is triggered  3- No exception is thrown | 2- A proactive command DISPLAY TEXT is fetched  TERMINAL RESPONSE |
| 7 | ProactiveResponseHandler availability with EVENT\_FORMATTED\_SMS\_CB  1- Envelope Cell Broadcast Download formatted is sent to the (U)SIM  Applet1 builds a proactive command DISPLAY TEXT  2- ProactiveHandler.send() method is called  3- ProactiveResponseHandler.getTheHandler() method is called. | 1- Applet1 is triggered  3- No exception is thrown | 2- A proactive command DISPLAY TEXT is fetched  TERMINAL RESPONSE |
| 8 | ProactiveResponseHandler availability with EVENT\_FORMATTED\_USSD  1- Envelope USSD formatted is sent to the (U)SIM  Applet builds a proactive command DISPLAY TEXT  2- ProactiveHandler.send() method is called  3- ProactiveResponseHandler.getTheHandler() method is called | 1- Applet1 is triggered  3- No exception is thrown | 2- A proactive command DISPLAY TEXT is fetched  TERMINAL RESPONSE |
| 9 | ProactiveResponseHandler availability with  EVENT\_UNFORMATTED\_USSD  1- Envelope USSD unformatted is sent to the (U)SIM  Applet1 builds a proactive command DISPLAY TEXT  2- ProactiveHandler.send() method is called  3- ProactiveResponseHandler.getTheHandler() method is called  4- Applet2 builds a proactive command DISPLAY TEXT  4- ProactiveHandler.send() method is called  5- ProactiveResponseHandler.getTheHandler() method is called | 1- Applet1 is triggered  3- No exception is thrown  Applet1 finalizes  Applet2 is triggered  5- No exception is thrown | 2- A proactive command DISPLAY TEXT is fetched  TERMINAL RESPONSE  4- A proactive command DISPLAY TEXT is fetched  TERMINA RESPONSE |
| 10 | ProactiveResponseHandler availability with EVENT\_DOWNLOAD\_IWLAN\_ACCESS\_STATUS  1- Envelope USSD unformatted is sent to the (U)SIM  Applet1 builds a proactive command DISPLAY TEXT  2- ProactiveHandler.send() method is called  3- ProactiveResponseHandler.getTheHandler() method is called  4- Applet2 builds a proactive command DISPLAY TEXT  4- ProactiveHandler.send() method is called  5- ProactiveResponseHandler.getTheHandler() method is called | 1- Applet1 is triggered  3- No exception is thrown  Applet1 finalizes  Applet2 is triggered | 2- A proactive command DISPLAY TEXT is fetched  TERMINAL RESPONSE  4- A proactive command DISPLAY TEXT is fetched  TERMINAL RESPONSE |
| 11 | The ProactiveHandler is not available before the Terminal Profile with EVENT\_FORMATTED\_SMS\_PP\_ENV  1- Reset the card without sending the Terminal Profile  2- Envelope SMS-PP Download formatted is sent is sent to the (U)SIM  3- Applet1 gets the ProactiveHandler  4- Applet1 gets the ProactiveResponseHandler | 2- Applet1 is triggered  3- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  4- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet1 finalizes |  |
| 12 | The ProactiveHandler is not available before the Terminal Profile with EVENT\_FORMATTED\_SMS\_PP\_UPD  1- Update Record EFSMS instruction formatted is sent to the (U)SIM  2- Applet1 gets the ProactiveHandler  3- Applet1 gets the ProactiveResponseHandler | 1- Applet1 is triggered  2- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  3- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet1 finalizes |  |
| 13 | The ProactiveHandler is not available before the Terminal Profile with EVENT\_UNFORMATTED\_SMS\_PP\_ENV  1- Envelope SMS-PP Download unformatted is sent to the (U)SIM  2- Applet1 gets the ProactiveHandler  3- Applet1 gets the ProactiveResponseHandler  4- Applet2 gets the ProactiveHandler  5- Applet2 gets the ProactiveResponseHandler | 1- Applet1 is triggered  2- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  3- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet1 finalizes  Applet2 is triggered  4- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  5- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet2 finalizes |  |
| 14 | The ProactiveHandler is not available before the Terminal Profile with EVENT\_UNFORMATTED\_SMS\_PP\_UPD  1- Update Record EFSMS instruction unformatted is sent to the (U)SIM  2- Applet1 gets the ProactiveHandler  3- Applet1 gets the ProactiveResponseHandler  4- Applet2 gets the ProactiveHandler  5- Applet2 gets the ProactiveResponseHandler | 1- Applet1 is triggered  2- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  3- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet1 finalizes  Applet2 is triggered  4- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  5- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet2 finalizes |  |
| 15 | The ProactiveHandler is not available before the Terminal Profile with EVENT\_UNFORMATTED\_SMS\_CB  1- Envelope Cell Broadcast Download unformatted is sent to the (U)SIM  2- Applet1 gets the ProactiveHandler  3- Applet1 gets the ProactiveResponseHandler  4- Applet2 gets the ProactiveHandler  5- Applet2 gets the ProactiveResponseHandler | 1- Applet1 is triggered  2- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  3- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet1 finalizes  Applet2 is triggered  4- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  5- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet2 finalizes |  |
| 16 | The ProactiveHandler is not available before the Terminal Profile with EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM  1- Envelope MO short message control by SIM is sent to the (U)SIM  2- Applet1 gets the ProactiveHandler  3- Applet1 gets the ProactiveResponseHandler | 1- Applet1 is triggered  2- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  3- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet1 finalizes |  |
| 17 | The ProactiveHandler is not available before the Terminal Profile with EVENT\_FORMATTED\_SMS\_CB  1- Envelope Cell Broadcast Download formatted is sent to the (U)SIM  2- Applet1 gets the ProactiveHandler  3- Applet1 gets the ProactiveResponseHandler | 1- Applet1 is triggered  2- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  3- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet1 finalizes |  |
| 18 | The ProactiveHandler is not available before the Terminal Profile with EVENT\_FORMATTED\_USSD  1- Envelope USSD formatted is sent is sent to the (U)SIM  2- Applet1 gets the ProactiveHandler  3- Applet1 gets the ProactiveResponseHandler | 1- Applet1 is triggered  2- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  3- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet1 finalizes |  |
| 19 | The ProactiveHandler is not available before the Terminal Profile with EVENT\_UNFORMATTED\_USSD  1- Envelope USSD unformatted is sent to the (U)SIM  2- Applet1 gets the ProactiveHandler  3- Applet1 gets the ProactiveResponseHandler  4- Applet2 gets the ProactiveHandler  5- Applet2 gets the ProactiveResponseHandler | 1- Applet1 is triggered  2- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  3- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet1 finalizes  Applet2 is triggered  4- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  5- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet2 finalizes |  |
| 20 | The ProactiveHandler is not available before the Terminal Profile with EVENT\_DOWNLOAD\_IWLAN\_ACCESS\_STATUS  1- Envelope Download Iwlan Access Status is sent to the (U)SIM  2- Applet1 gets the ProactiveHandler  3- Applet1 gets the ProactiveResponseHandler  4- Applet2 gets the ProactiveHandler  5- Applet2 gets the ProactiveResponseHandler | 1- Applet1 is triggered  2- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  3- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet1 finalizes  Applet2 is triggered  4- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  5- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet2 finalizes |  |

#### 5.3.1.3 EnvelopeHandler

Test Area Reference: Ufw\_Mha\_Enhd

5.3.1.3.1 Conformance requirements

5.3.1.3.1.1 Normal execution

- CRRN1: The *EnvelopeHandler* and its content are available for all toolkit applets triggered from the invocation to the termination of their *processToolkit()* method for the following events:

EVENT\_FORMATTED\_SMS\_PP\_ENV

EVENT\_FORMATTED\_SMS\_PP\_UPD

EVENT\_UNFORMATTED\_SMS\_PP\_ENV

EVENT\_UNFORMATTED\_SMS\_PP\_UPD

EVENT\_UNFORMATTED\_SMS\_CB

EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM

EVENT\_FORMATTED\_SMS\_CB

EVENT\_FORMATTED\_USSD

EVENT\_UNFORMATTED\_USSD

EVENT\_DOWNLOAD\_IWLAN\_ACCESS

- CRRN2: An *EnvelopeHandler* is considered available when no HANLDER\_NOT\_AVAILABLE ToolkitException is thrown when the corresponding *getTheHandler()* method is called or a method of the handler is called.

5.3.1.3.1.2 Parameter errors

No requirements.

5.3.1.3.1.3 Context Errors

No requirements.

5.3.1.3.2 Test area files

Test Source: Test\_Ufw\_Mha\_Enhd.java

Test Applet: Ufw\_Mha\_Enhd\_1.java

Ufw\_Mha\_Enhd\_2.java

Cap File: Ufw\_Mha\_Enhd.cap

5.3.1.3.3 Test coverage

| CRR Number | Test Case Number |
| --- | --- |
| N1 | 1 to 10 |
| N2 | 1 to 10 |

5.3.1.3.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
| 1 | EnvelopeHandler availability with EVENT\_FORMATTED\_SMS\_PP\_ENV  1- Envelope SMS-PP Download formatted is sent to the (U)SIM  2- EnvelopeHandler.getTheHandler() method is called by Applet1 | 1- Applet1 is triggered  2- No exception is thrown. |  |
| 2 | EnvelopeHandler availability with EVENT\_FORMATTED\_SMS\_PP\_UPD  1- Update Record EFSMS instruction formatted is sent to the (U)SIM  2- EnvelopeHandler.getTheHandler() method is called by Applet1 | 1- Applet1 is triggered  2- No exception is thrown. |  |
| 3 | EnvelopeHandler availability with EVENT\_UNFORMATTED\_SMS\_PP\_ENV  1- Envelope SMS-PP Download unformatted is sent to the (U)SIM  2- EnvelopeHandler.getTheHandler() method is called by Applet1  3- EnvelopeHandler.getTheHandler() method is called by Applet2 | 1- Applet1 is triggered  2- No exception is thrown.  Applet1 finalizes  3- Applet2 is triggered  4- No exception is thrown. |  |
| 4 | EnvelopeHandler availability with EVENT\_UNFORMATTED\_SMS\_PP\_UPD  1- Update Record EFSMS instruction unformatted is sent to the (U)SIM  2- EnvelopeHandler.getTheHandler() method is called by Applet1  3- EnvelopeHandler.getTheHandler() method is called by Applet2 | 1- Applet1 is triggered  2- No exception is thrown.  Applet1 finalizes  Applet2 is triggered  3- No exception is thrown. |  |
| 5 | EnvelopeHandler availability with EVENT\_UNFORMATTED\_SMS\_CB  1- Envelope Cell Broadcast Download unformatted is sent to the (U)SIM  2- EnvelopeHandler.getTheHandler() method is called by Applet1  3- EnvelopeHandler.getTheHandler() method is called by Applet2 | 1- Applet1 is triggered  2- No exception is thrown  Applet1 finalizes  3- Applet2 is triggered  4- No exception is thrown |  |
| 6 | EnvelopeHandler availability with EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM  1- Envelope MO short message control by SIM is sent to the (U)SIM  2- EnvelopeHandler.getTheHandler() method is called by Applet1 | 1- Applet1 is triggered  2- No exception is throw |  |
| 7 | EnvelopeHandler availability with EVENT\_FORMATTED\_SMS\_CB  1- Envelope Cell Broadcast Download formatted is sent to the (U)SIM  2- EnvelopeHandler.getTheHandler() method is called by Applet1 | 1- Applet1 is triggered  2-No exception is thrown |  |
| 8 | EnvelopeHandler availability with EVENT\_FORMATTED\_USSD  1- Envelope USSD formatted is sent to the (U)SIM  2- EnvelopeHandler.getTheHandler() method is called by Applet1 | 1- Applet1 is triggered  2-No exception is thrown |  |
| 9 | EnvelopeHandler availability with EVENT\_UNFORMATTED\_USSD  1- Envelope USSD unformatted is sent to the (U)SIM  2- EnvelopeHandler.getTheHandler() method is called by Applet1  3- EnvelopeHandler.getTheHandler() method is called by Applet2 | 1- Applet1 is triggered  2- No exception is thrown.  Applet1 finalizes  3- Applet2 is triggered  4- No exception is thrown |  |
| 10 | EnvelopeHandler availability with DOWNLOAD\_IWLAN\_ACCESS\_STATUS  1- Envelope Download Iwlan Access Status is sent to the (U)SIM  2- EnvelopeHandler.getTheHandler() method is called by Applet1  3- EnvelopeHandler.getTheHandler() method is called by Applet2 | 1- Applet1 is triggered  2- No exception is thrown.  Applet1 finalizes  3- Applet2 is triggered  4- No exception is thrown |  |

#### 5.3.1.4 EnvelopeResponseHandler

Test Area Reference: Ufw\_Mha\_Erhd

5.3.1.4.1 Conformance requirements

5.3.1.4.1.1 Normal execution

- CRRN1: The handler is available for all triggered toolkit applets from the invocation of the *processToolkit()* method of the toolkit applet until a toolkit applet has posted an envelope response or the first invocation of the *ProactiveHandler.send()* method for the following events:

EVENT\_FORMATTED\_SMS\_PP\_ENV

EVENT\_UNFORMATTED\_SMS\_PP\_ENV

EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM

EVENT\_FORMATTED\_USSD

EVENT\_UNFORMATTED\_USSD

- CRRN2: An *EnvelopeResponseHandler* is considered available when no HANLDER\_NOT\_AVAILABLE ToolkitException is thrown when the corresponding *getTheHandler()* method is called or a method of the handler is called.

5.3.1.4.1.2 Parameter errors

No requirements.

5.3.1.4.1.3 Context Errors

- CRRC1: The handler is not available for the following events:

EVENT\_FORMATTED\_SMS\_PP\_UPD

EVENT\_UNFORMATTED\_SMS\_PP\_UPD

EVENT\_UNFORMATTED\_SMS\_CB

EVENT\_FORMATTED\_SMS\_CB

EVENT\_DOWNLOAD\_IWLAN\_ACCESS\_STATUS

5.3.1.4.2 Test area files

Test Source: Test\_Ufw\_Mha\_Erhd.java

Test Applet: Ufw\_Mha\_Erhd\_1.java

Ufw\_Mha\_Erhd\_2.java

Cap File: Ufw\_Mha\_Erhd.cap

5.3.1.4.3 Test coverage

| CRR Number | Test Case Number |
| --- | --- |
| N1 | 5 to 10 |
| N2 | 1 to 10 |
| C1 | 1 to 11 |

5.3.1.4.4 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| 1 | EnvelopeResponseHandler availability with EVENT\_FORMATTED\_SMS\_PP\_UPD  1- Update Record EFSMS instruction formatted is sent to the (U)SIM  2- EnvelopeResponseHandler.getTheHandler() method is called by Applet1 | 1- The applet1 is triggered.  2- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown |  |
| 2 | EnvelopeResponseHandler availability with EVENT\_UNFORMATTED\_SMS\_PP\_UPD  1- Update Record EFSMS instruction unformatted is sent to the (U)SIM  2- EnvelopeResponseHandler.getTheHandler() method is called by Applet1 | 1- Applet1 is triggered.  2- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown |  |
| 3 | EnvelopeResponseHandler availability with EVENT\_FORMATTED\_SMS\_CB  1- Envelope Cell Broadcast Download formatted is sent to the SIM  2- EnvelopeResponseHandler.getTheHandler() method is called by Applet1 | 1- The applet1 is triggered.  2- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown |  |
| 4 | EnvelopeResponseHandler availability with EVENT\_UNFORMATTED\_SMS\_CB  1- Envelope Cell Broadcast Download unformatted is sent to the SIM  2- EnvelopeResponseHandler.getTheHandler() method is called by Applet1 | 1- Applet1 is triggered.  2- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown |  |
| 5 | EnvelopeResponseHandler availability with EVENT\_FORMATTED\_SMS\_PP\_ENV  1- Envelope SMS-PP Download formatted is sent to the (U)SIM  2- EnvelopeResponseHandler.getTheHandler() method is called by Applet1  3- Applet1 builds an additional information for response packet and it calls the post() method  4- Applet1 calls all methods of the EnvelopeResponseHandler (including inherited methods)  5- A EVENT\_FORMATTED\_SMS\_PP\_ENV envelope is sent to the (U)SIM  6- EnvelopeResponseHandler.getTheHandler() method is called by Applet1  7- Applet1 builds a proactive command and it calls the send() method  8- Applet1 calls all methods of the EnvelopeResponseHandler (including inherited methods) | 1- Applet1 is triggered  2- No exception is thrown.  4- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown for each method  Applet1 finalizes  5- Applet1 is triggered  6- No Exception is thrown  8- ToolkitException HANDLER\_NOT\_AVAILABLE is thrown for each method | 3- The response packet is sent  7- The proactive command is sent |
| 6 | EnvelopeResponseHandler availability with EVENT\_UNFORMATTED\_SMS\_PP\_ENV  1- Envelope SMS-PP Download unformatted is sent to the (U)SIM  2- EnvelopeResponseHandler.getTheHandler() method is called by Applet1  3- Applet1 builds the envelope response and it calls the post() method  4- Applet1 calls all methods of the EnvelopeResponseHandler (including inherited methods)  5- EnvelopeResponseHandler.getTheHandler() method is called  6- An unformatted SMS PP envelope is sent to the (U)SIM  7- EnvelopeResponseHandler.getTheHandler() method is called.  8- Applet1 builds a proactive command and it calls the send() method  9- Applet1 calls all methods of the EnvelopeResponseHandler (including inherited methods)  10- EnvelopeResponseHandler.getTheHandler() method is called by Applet2 | 1- Applet1 is triggered  2- No exception is thrown.  4- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown for each method  Applet1 finalizes  5- Applet2 is triggered.  A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown.  Applet2 finalizes  6- Applet1 is triggered.  7- No exception is thrown.  9- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown for each method.  Applet1 finalizes  10- Applet2 is triggered.  A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown. | 3- The envelope response is sent  9- The proactive command is fetched and the Terminal response is issued. |
| 7 | EnvelopeResponseHandler availability with EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM  1- Envelope MO short message control by SIM is sent to the (U)SIM  2- EnvelopeResponseHandler.getTheHandler() method is called by Applet1  3- Applet1 builds the envelope response and it calls the postAsBERTLV() method  4- Applet1 calls all methods of the EnvelopeResponseHandler (including inherited methods)  5- Envelope MO short message control by SIM is sent to the (U)SIM  6- EnvelopeResponseHandler.getTheHandler() method is called by Applet1  7- Applet1 builds a proactive command and it calls the send method  8- Applet1 calls all methods of the EnvelopeResponseHandler (including inherited methods) | 1- Applet1 is triggered  2- No exception is thrown.  4- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown for each method  Applet1 finalizes  5- Applet1 is triggered  6- No exception is thrown  8- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown for each method | 3-The envelope response is sent  7- The proactive command is fetched and the Terminal Response is issued |
| 8 | EnvelopeResponseHandler availability with EVENT\_UNFORMATTED\_SMS\_PP\_ENV in case of multi-triggering  1- Envelope SMS-PP Download unformatted is sent to the (U)SIM  2- EnvelopeResponseHandler.getTheHandler() method is called by Applet1  5- EnvelopeResponseHandler.getTheHandler() method is called by Applet 2  6- Applet2 calls the post() method | 1- Applet1 is triggered  2- No exception is thrown.  3- Applet1 finalizes  4- Applet2 is triggered.  5- No Exception is thrown  Applet2 finalizes | 6- The response is checked. |
| 9 | EnvelopeResponseHandler availability with EVENT\_FORMATTED\_USSD  1- Envelope USSD formatted is sent to the (U)SIM  2- EnvelopeResponseHandler.getTheHandler() method is called by Applet1  3- Applet1 builds an additional information for response packet and it calls the post() method  4- Applet1 calls all methods of the EnvelopeResponseHandler (including inherited methods)  5- EVENT\_FORMATTED\_USSD envelope is sent to the (U)SIM  6- EnvelopeResponseHandler.getTheHandler() method is called by Applet1  7- Applet1 builds a proactive command and it calls the send() method  8- Applet1 calls all methods of the EnvelopeResponseHandler (including inherited methods) | 1- Applet1 is triggered  2- No exception is thrown.  4- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown for each method  Applet1 finalizes  5- Applet1 is triggered  6- No Exception is thrown  8- ToolkitException HANDLER\_NOT\_AVAILABLE is thrown for each method | 3- The response packet is sent  7- The proactive command is sent |
| 10 | EnvelopeResponseHandler availability with EVENT\_UNFORMATTED\_SMS\_PP\_ENV in case of multi-triggering  1- Envelope USSD unformatted is sent to the (U)SIM  2- EnvelopeResponseHandler.getTheHandler() method is called by Applet1  5- EnvelopeResponseHandler.getTheHandler() method is called by Applet 2  6- Applet2 calls the post() method | 1- Applet1 is triggered  2- No exception is thrown.  3- Applet1 finalizes  4- Applet2 is triggered.  5- No Exception is thrown  Applet2 finalizes | 6- The response is checked. |
| 11 | EnvelopeResponseHandler availability with EVENT\_DOWNLOAD\_IWLAN\_ACCESS\_STATUS  1- Envelope Download Iwlan Access Status is sent to the (U)SIM  2- EnvelopeResponseHandler.getTheHandler() method is called by Applet1 | 1- The applet1 is triggered.  2- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown |  |

#### 5.3.1.5 USATEnvelopeHandler

Test Area Reference: Ufw\_Mha\_Uehd

5.3.1.5.1 Conformance requirements

5.3.1.5.1.1 Normal execution

- CRRN1: The *UsatEnvelopeHandler* and its content are available for all toolkit applets triggered from the invocation to the termination of their *processToolkit()* method for the following events:

EVENT\_FORMATTED\_SMS\_PP\_ENV

EVENT\_FORMATTED\_SMS\_PP\_UPD

EVENT\_UNFORMATTED\_SMS\_PP\_ENV

EVENT\_UNFORMATTED\_SMS\_PP\_UPD

EVENT\_UNFORMATTED\_SMS\_CB

EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM

EVENT\_FORMATTED\_SMS\_CB

EVENT\_FORMATTED\_USSD

EVENT\_UNFORMATTED\_USSD

EVENT\_DOWNLOAD\_IWLAN\_ACCESS\_STATUS

- CRRN2: An *UsatEnvelopeHandler* is considered available when no HANLDER\_NOT\_AVAILABLE ToolkitException is thrown when the corresponding *getTheHandler()* method is called or a method of the handler is called.

5.3.1.5.1.2 Parameter errors

No requirements.

5.3.1.5.1.3 Context Errors

No requirements.

5.3.1.5.2 Test area files

Test Source: Test\_Ufw\_Mha\_Uehd.java

Test Applet: Ufw\_Mha\_Uehd\_1.java

Ufw\_Mha\_Uehd\_2.java

Cap File: Ufw\_Mha\_Uehd.cap

5.3.1.5.3 Test coverage

| CRR Number | Test Case Number |
| --- | --- |
| N1 | 1 to 10 |
| N2 | 1 to 10 |

5.3.1.5.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
| 1 | USATEnvelopeHandler availability with EVENT\_FORMATTED\_SMS\_PP\_ENV  1- Envelope SMS-PP Download formatted is sent to the (U)SIM  2- USATEnvelopeHandler.getTheHandler() method is called by Applet1 | 1- Applet1 is triggered  2- No exception is thrown. |  |
| 2 | USATEnvelopeHandler availability with EVENT\_FORMATTED\_SMS\_PP\_UPD  1- Update Record EFSMS instruction formatted is sent to the (U)SIM  2- USATEnvelopeHandler.getTheHandler() method is called by Applet1 | 1- Applet1 is triggered  2- No exception is thrown. |  |
| 3 | USATEnvelopeHandler availability with EVENT\_UNFORMATTED\_SMS\_PP\_ENV  1- Envelope SMS-PP Download unformatted is sent to the (U)SIM  2- USATEnvelopeHandler.getTheHandler() method is called by Applet1  3- USATEnvelopeHandler.getTheHandler() method is called by Applet2 | 1- Applet1 is triggered  2- No exception is thrown.  Applet1 finalizes  3- Applet2 is triggered  4- No exception is thrown. |  |
| 4 | USATEnvelopeHandler availability with EVENT\_UNFORMATTED\_SMS\_PP\_UPD  1- Update Record EFSMS instruction unformatted is sent to the (U)SIM  2- USATEnvelopeHandler.getTheHandler() method is called by Applet1  3- USATEnvelopeHandler.getTheHandler() method is called by Applet2 | 1- Applet1 is triggered  2- No exception is thrown.  Applet1 finalizes  Applet2 is triggered  3- No exception is thrown. |  |
| 5 | USATEnvelopeHandler availability with EVENT\_UNFORMATTED\_SMS\_CB  1- Envelope Cell Broadcast Download unformatted is sent to the (U)SIM  2- USATEnvelopeHandler.getTheHandler() method is called by Applet1  3- USATEnvelopeHandler.getTheHandler() method is called by Applet2 | 1- Applet1 is triggered  2- No exception is thrown  Applet1 finalizes  3- Applet2 is triggered  4- No exception is thrown |  |
| 6 | USATEnvelopeHandler availability with EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM  1- Envelope MO short message control by SIM is sent to the (U)SIM  2- USATEnvelopeHandler.getTheHandler() method is called by Applet1 | 1- Applet1 is triggered  2- No exception is throw |  |
| 7 | USATEnvelopeHandler availability with EVENT\_FORMATTED\_SMS\_CB  1- Envelope Cell Broadcast Download formatted is sent to the (U)SIM  2- USATEnvelopeHandler.getTheHandler() method is called by Applet1 | 1- Applet1 is triggered  2-No exception is thrown |  |
| 8 | USATEnvelopeHandler availability with EVENT\_FORMATTED\_USSD  1- Envelope USSD formatted is sent to the (U)SIM  2- USATEnvelopeHandler.getTheHandler() method is called by Applet1 | 1- Applet1 is triggered  2- No exception is thrown. |  |
| 9 | USATEnvelopeHandler availability with EVENT\_UNFORMATTED\_USSD  1- Envelope USSD unformatted is sent to the (U)SIM  2- USATEnvelopeHandler.getTheHandler() method is called by Applet1  3- USATEnvelopeHandler.getTheHandler() method is called by Applet2 | 1- Applet1 is triggered  2- No exception is thrown.  Applet1 finalizes  3- Applet2 is triggered  4- No exception is thrown. |  |
| 10 | USATEnvelopeHandler availability with EVENT\_DOWNLOAD\_IWLAN\_ACCESS\_STATUS  1- Envelope USSD unformatted is sent to the (U)SIM  2- USATEnvelopeHandler.getTheHandler() method is called by Applet1  3- USATEnvelopeHandler.getTheHandler() method is called by Applet2 | 1- Applet1 is triggered  2- No exception is thrown.  Applet1 finalizes  3- Applet2 is triggered  4- No exception is thrown |  |

#### 5.3.1.6 Applet triggering with ongoing proactive session

Test Area Reference: Ufw\_Mha\_Rent

5.3.1.6.1 Conformance requirements

5.3.1.6.1.1 Normal execution

- CRRN1: *EnvelopeHandler* and *USATEnvelopeHandler* are available for all events.

- CRRN2: *EnvelopeResponseHandler* is available for the following events:

EVENT\_FORMATTED\_SMS\_PP\_ENV

EVENT\_UNFORMATTED\_SMS\_PP\_ENV

EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM

EVENT\_FORMATTED\_USSD

EVENT\_UNFORMATTED\_USSD

- CRRN3: Reply busy is not allowed for following events:

EVENT\_FORMATTED\_SMS\_PP\_UPD

EVENT\_UNFORMATTED\_SMS\_PP\_UPD

EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM

5.3.1.6.1.2 Parameter errors

No requirements.

5.3.1.6.1.3 Context Errors

- CRRC1: EnvelopeResponseHandler is not available for the following events:

EVENT\_FORMATTED\_SMS\_PP\_UPD

EVENT\_UNFORMATTED\_SMS\_PP\_UPD

EVENT\_UNFORMATTED\_SMS\_CB

EVENT\_FORMATTED\_SMS\_CB

EVENT\_DOWNLOAD\_IWLAN\_ACCESS\_STATUS

5.3.1.6.2 Test area files

Test Source: Test\_Ufw\_Mha\_Rent.java

Test Applet: Ufw\_Mha\_Rent\_1.java

Cap File: Ufw\_Mha\_Rent.cap

5.3.1.6.3 Test coverage

| CRR Number | Test Case Number |
| --- | --- |
| N1 | 1, 2, 3, 4, 5, 6, 7, 8 |
| N2 | 3, 4, 5, 8, 9 |
| N3 | 1, 2, 3 |
| C1 | 1, 2, 6, 7, 10 |

5.3.1.6.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
| 1 | Handlers availability with EVENT\_FORMATTED\_SMS\_PP\_UPD  1- Envelope unrecognized is sent to the (U)SIM  2- Applet1 builds a proactive command and calls the send() method  3- Update Record EFSMS instruction formatted is sent to the (U)SIM  4- EnvelopeHandler.getTheHandler() and USATEnvelopeHandler.getTheHandler() methods are called by Applet1  5- EnvelopeResponseHandler.getTheHandler() method is called by Applet1 | 1- Applet1 is triggered  3- Applet1 is triggered again  4- No exception is thrown  5- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet1 finalizes | 2- 91 XX  6- The proactive command is fetched and the Terminal Response is issued. |
| 2 | Handlers availability with EVENT\_UNFORMATTED\_SMS\_PP\_UPD  1- Envelope unrecognized is sent to the (U)SIM  2- Applet1 builds a proactive command and calls the send() method  3- Update Record EFSMS instruction unformatted is sent to the (U)SIM  4- EnvelopeHandler.getTheHandler() and USATEnvelopeHandler.getTheHandler() methods are called by Applet1  5- EnvelopeResponseHandler.getTheHandler() method is called by Applet1 | 1- Applet1 is triggered  3- Applet1 is triggered again  4- No exception is thrown  5- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet1 finalizes | 2- 91 XX  6- The proactive command is fetched and the Terminal Response is issued. |
| 3 | Handlers availability with EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM  1- Envelope unrecognized is sent to the (U)SIM  2- Applet1 builds a proactive command and calls the send() method  3- Envelope MO short message control by SIM is sent to the (U)SIM  4- EnvelopeHandler.getTheHandler() and USATEnvelopeHandler.getTheHandler() methods are called by Applet1  5- EnvelopeResponseHandler.getTheHandler() method is called by Applet1 | 1- Applet1 is triggered  3- Applet1 is triggered again  4- No exception is thrown  5- No exception is thrown  Applet1 finalizes | 2- 91 XX  6- The proactive command is fetched and the Terminal Response is issued. |
| 4 | Handlers availability with EVENT\_FORMATTED\_SMS\_PP\_ENV  1- Envelope unrecognized is sent to the (U)SIM  2- Applet1 builds a proactive command and calls the send() method  3- Envelope SMS-PP Download formatted is sent to the (U)SIM  4- EnvelopeHandler.getTheHandler() and USATEnvelopeHandler.getTheHandler() methods are called by Applet1  5- EnvelopeResponseHandler.getTheHandler() method is called by Applet1 | 1- Applet1 is triggered  3- Applet1 is triggered again  4- No exception is thrown  5- No exception is thrown  Applet1 finalizes | 2- 91 XX  6- The proactive command is fetched and the Terminal Response is issued. |
| 5 | Handlers availability with EVENT\_UNFORMATTED\_SMS\_PP\_ENV  1- Envelope unrecognized is sent to the (U)SIM  2- Applet1 builds a proactive command and calls the send() method  3- Envelope SMS-PP Download unformatted is sent to the (U)SIM  4- EnvelopeHandler.getTheHandler() and USATEnvelopeHandler.getTheHandler() methods are called by Applet1  5- EnvelopeResponseHandler.getTheHandler() method is called by Applet1 | 1- Applet1 is triggered  3- Applet1 is triggered again  4- No exception is thrown  5- No exception is thrown  Applet1 finalizes | 2- 91 XX  6- The proactive command is fetched and the Terminal Response is issued. |
| 6 | Handlers availability with EVENT\_FORMATTED\_SMS\_CB  1- Envelope unrecognized is sent to the (U)SIM  2- Applet1 builds a proactive command and calls the send() method  3- Envelope CB Download formatted is sent to the (U)SIM  4- EnvelopeHandler.getTheHandler() and USATEnvelopeHandler.getTheHandler() methods are called by Applet1  5- EnvelopeResponseHandler.getTheHandler() method is called by Applet1 | 1- Applet1 is triggered  3- Applet1 is triggered again  4- No exception is thrown  5- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet1 finalizes | 2- 91 XX  6- The proactive command is fetched and the Terminal Response is issued. |
| 7 | Handlers availability with EVENT\_UNFORMATTED\_SMS\_CB  1- Envelope unrecognized is sent to the (U)SIM  2- Applet1 builds a proactive command and calls the send() method  3- Envelope CB Download unformatted is sent to the (U)SIM  4- EnvelopeHandler.getTheHandler() and USATEnvelopeHandler.getTheHandler() methods are called by Applet1  5- EnvelopeResponseHandler.getTheHandler() method is called by Applet1 | 1- Applet1 is triggered  3- Applet1 is triggered again  4- No exception is thrown  5- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet1 finalizes | 2- 91 XX  6- The proactive command is fetched and the Terminal Response is issued. |
| 8 | Handlers availability with EVENT\_FORMATTED\_SMS\_USSD  1- Envelope unrecognized is sent to the (U)SIM  2- Applet1 builds a proactive command and calls the send() method  3- Envelope USSD formatted is sent to the (U)SIM  4- EnvelopeHandler.getTheHandler() and USATEnvelopeHandler.getTheHandler() methods are called by Applet1  5- EnvelopeResponseHandler.getTheHandler() method is called by Applet1 | 1- Applet1 is triggered  3- Applet1 is triggered again  4- No exception is thrown  5- No exception is thrown  Applet1 finalizes | 2- 91 XX  6- The proactive command is fetched and the Terminal Response is issued. |
| 9 | Handlers availability with EVENT\_UNFORMATTED\_USSD  1- Envelope unrecognized is sent to the (U)SIM  2- Applet1 builds a proactive command and calls the send() method  3- Envelope USSD unformatted is sent to the (U)SIM  4- EnvelopeHandler.getTheHandler() and USATEnvelopeHandler.getTheHandler() methods are called by Applet1  5- EnvelopeResponseHandler.getTheHandler() method is called by Applet1 | 1- Applet1 is triggered  3- Applet1 is triggered again  4- No exception is thrown  5- No exception is thrown  Applet1 finalizes | 2- 91 XX  6- The proactive command is fetched and the Terminal Response is issued. |
| 10 | Handlers availability with EVENT\_DOWNLOAD\_IWLAN\_ACCESS\_STATUS  1- Envelope unrecognized is sent to the (U)SIM  2- Applet1 builds a proactive command and calls the send() method  3- Envelope Download Iwlan Access Status is sent to the (U)SIM  4- EnvelopeHandler.getTheHandler() and USATEnvelopeHandler.getTheHandler() methods are called by Applet1  5- EnvelopeResponseHandler.getTheHandler() method is called by Applet1 | 1- Applet1 is triggered  3- Applet1 is triggered again  4- No exception is thrown  5- A ToolkitException HANDLER\_NOT\_AVAILABLE is thrown  Applet1 finalizes | 2- 91 XX  6- The proactive command is fetched and the Terminal Response is issued. |

### 5.3.2 Handler integrity

#### 5.3.2.1 ProactiveResponseHandler

Test Area Reference: Ufw\_Hin\_Prhd

5.3.2.1.1 Conformance requirements

5.3.2.1.1.1 Normal execution

- CRRN1: The *ProactiveResponseHandler* TLV list shall be empty before the first call to the *ProactiveHandler.send()* method.

5.3.2.1.1.2 Parameter errors

No requirements.

5.3.2.1.1.3 Context Errors

No requirements.

5.3.2.1.2 Test area files

Test Source: Test\_Ufw\_Hin\_Prhd.java

Test Applet: Ufw\_Hin\_Prhd\_1.java

Cap File: Ufw\_Hin\_Prhd.cap

5.3.2.1.3 Test coverage

| CRR Number | Test Case Number |
| --- | --- |
| N1 | 1 |

5.3.2.1.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
| 1 | Applet registration and ProactiveResponseHandler obtaining  1- Applet is registered to all events defined in TS 31.130 [2].  Using the method setEventList() for all the events.  Terminal Profile command is sent to the (U)SIM without the facilities of SET\_EVENT\_LIST ,SETUP\_IDLE\_MODE\_TEXT, SETUP\_MENU and POLL\_INTERVAL.  2- For each event/triggering:  3- ProactiveResponseHandler.getTheHandler() is called  4- ProactiveResponseHandler.getLength() is called | 1- No exception is thrown  2- Applet is triggered.  3- No exception is thrown  4- The return value is 0 |  |

#### 5.3.2.2 EnvelopeHandler

Test Area Reference: Ufw\_Hin\_Enhd

5.3.2.2.1 Conformance requirements

5.3.2.2.1.1 Normal execution

- CRRN1: When available, the *EnvelopeHandler* shall remain available and its content shall remain unchanged from the invocation to the termination of the *processToolkit()* method.

- CRRN2: The *EnvelopeHandler* TLV list is filled with the Comprehension TLV data objects of the ENVELOPE APDU command. The Comprehension TLV data objects shall be provided in the order given in the ENVELOPE command data.

5.3.2.2.1.2 Parameter errors

No requirements.

5.3.2.2.1.3 Context Errors

No requirements.

5.3.2.2.2 Test area files

Test Source: Test\_Ufw\_Hin\_Enhd.java

Test Applet: Ufw\_Hin\_Enhd\_1.java

Cap File: Ufw\_Hin\_Enhd.cap

5.3.2.2.3 Test coverage

| CRR Number | Test Case Number |
| --- | --- |
| N1 | 1 to 10 |
| N2 | 1 to 10 |

5.3.2.2.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
| 1 | EnvelopeHandler integrity checks with EVENT\_FORMATTED\_SMS\_PP\_ENV  1- A formatted SMS PP envelope is sent to the (U)SIM  2- EnvelopeHandler.getTheHandler() method is called  3- Copy the content of the EnvelopeHandler in buffer1 using EnvelopeHandler.copy()  The EnvelopeHandler.findTLV() method is called with TAG\_SMS\_TPDU  4- A proactive command DISPLAY TEXT is sent  5- Envelope call control by SIM is sent to the (U)SIM  EnvelopeHandler.getTheHandler() method is called  6- Check that EnvelopeHandler content is the same as in the envelope call control using EnvelopeHandler.copy() and Util.arrayCompare() methods  The EnvelopeHandler.findTLV() method is called with TAG\_DEVICE\_IDENTITIES  Call Control execution is finished.  Check that the TAG\_SMS\_TPDU is the TLV selected  7- The content of EnvelopeHandler is compared with buffer1 using Util.arrayCompare() | 1- Applet is triggered  2- No exception is thrown.  3- No exception is thrown.  5- Applet is triggered  6- No exception is thrown and the handler contains the envelope call control by SIM  7- The contents of the EnvelopeHandler shall be the same as stored in buffer1 | 4- 91 XX  Proactive command Display Text is fetched  The terminal Response of DISPLAY TEXT is sent to the (U)SIM |
| 2 | EnvelopeHandler integrity checks with EVENT\_FORMATTED\_SMS\_PP\_UPD  1- Update Record EFSMS instruction single and formatted is sent to the (U)SIM  2- EnvelopeHandler.getTheHandler() method is called  3- Copy the content of the EnvelopeHandler in buffer1 using EnvelopeHandler.copy()  The EnvelopeHandler.findTLV() method is called with TAG\_SMS\_TPDU  4- A proactive command DISPLAY TEXT is sent  5- Envelope call control by SIM is sent to the (U)SIM  EnvelopeHandler.getTheHandler() method is called  6- Checked that EnvelopeHandler content is the same as in envelope call control using EnvelopeHandler.copy() and Util.arrayCompare() methods  The EnvelopeHandler.findTLV() method is called with TAG\_SMS\_TPDU  Call Control execution is finished.  Checked that the TAG\_SMS\_TPDU is the TLV selected  7- The content of EnvelopeHandler is compared with buffer1 using Util.arrayCompare() | 1- Applet is triggered  2- No exception is thrown.  3- No exception is thrown.  5- Applet is triggered  6- No exception is thrown and the handler contains the envelope call control by SIM  7- The contents of the EnvelopeHandler shall be the same as stored in buffer1 | 4- 91 XX  Proactive command Display Text is fetched  The terminal Response of DISPLAY TEXT is sent to the (U)SIM |
| 3 | EnvelopeHandler integrity checks with EVENT\_UNFORMATTED\_SMS\_PP\_ENV  1- A unformatted SMS PP envelope is sent to the (U)SIM  2- EnvelopeHandler.getTheHandler() method is called  3- Copy the content of the EnvelopeHandler in buffer1 using EnvelopeHandler.copy()  The EnvelopeHandler.findTLV method is called with TAG\_DEVICE\_IDENTITIES  4- A proactive command DISPLAY TEXT is sent  5- Envelope call control by SIM is sent to the (U)SIM  EnvelopeHandler.getTheHandler() method is called  6- Check that EnvelopeHandler content is the same as in the envelope call control using EnvelopeHandler.copy() and Util.arrayCompare() methods  The EnvelopeHandler.findTLV() method is called with TAG\_DEVICE\_IDENTITIES  Call Control execution is finished.  Check that the TAG\_DEVICE\_IDENTITIES is the TLV selected  7- The content of EnvelopeHandler is compared with buffer1 using Util.arrayCompare() | 1- Applet is triggered  2- No exception is thrown.  3- No exception is thrown.  5- Applet is triggered  6- No exception is thrown and the handler contains the envelope call control by SIM  7- The contents of the EnvelopeHandler shall be the same as stored in buffer1. | 4- 91 XX  Proactive command Display Text is fetched  The terminal Response of DISPLAY TEXT is sent to the (U)SIM |
| 4 | EnvelopeHandler integrity checks with EVENT\_UNFORMATTED\_SMS\_PP\_UPD  1- Update Record EFSMS instruction single and unformatted is sent to the (U)SIM  2- EnvelopeHandler.getTheHandler() method is called  3- Copy the content of the EnvelopeHandler in buffer1 using EnvelopeHandler.copy()  The EnvelopeHandler.findTLV method is called with TAG\_SMS\_TPDU  4- A proactive command DISPLAY TEXT is sent  5- Envelope call control by SIM is sent to the (U)SIM  EnvelopeHandler.getTheHandler() method is called  6- Check that EnvelopeHandler content is the same as in envelope call control using EnvelopeHandler.copy() and Util.arrayCompare() methods  The EnvelopeHandler.findTLV() method is called with TAG\_DEVICE\_IDENTITIES  Call Control execution is finished.  Check that the SMS\_TPDU is the TLV selected  7- The content of EnvelopeHandler is compared with buffer1 using Util.arrayCompare() | 1- Applet is triggered  2- No exception is thrown.  3- No exception is thrown.  5- Applet is triggered  6- No exception is thrown and the handler contains the envelope call control by SIM  7- The contents of the EnvelopeHandler shall be the same as stored in buffer1. | 4- 91 XX  Proactive command Display Text is fetched  The terminal Response of DISPLAY TEXT is sent to the (U)SIM |
| 5 | EnvelopeHandler integrity checks with EVENT\_UNFORMATTED\_SMS\_CB  1- An unformatted cell broadcast envelope is sent to the (U)SIM  2- EnvelopeHandler.getTheHandler() method is called  3- Copy the content of the EnvelopeHandler in buffer1 using EnvelopeHandler.copy()  The EnvelopeHandler.findTLV() method is called with TAG\_CELLBROADCAST\_PAGE  4- A proactive command DISPLAY TEXT is sent  5- Envelope call control by SIM is sent to the (U)SIM  EnvelopeHandler.getTheHandler() method is called  6- Checked that EnvelopeHandler content is the same as in envelope call control using EnvelopeHandler.copy() and Util.arrayCompare() methods  The EnvelopeHandler.findTLV() method is called with TAG\_DEVICE\_IDENTITIES  Call Control execution is finished.  Check that the TAG\_CELLBROADCAST\_PAGE is the TLV selected  7- The content of EnvelopeHandler is compared with buffer1 using Util.arrayCompare() | 1- Applet is triggered  2- No exception is thrown.  3- No exception is thrown.  5- Applet is triggered  6- No exception is thrown and the handler contains the envelope call control by SIM  7- The contents of the EnvelopeHandler shall be the same as stored in buffer1. | 4- 91 XX  Proactive command Display Text is fetched  The terminal Response of DISPLAY TEXT is sent to the (U)SIM |
| 6 | EnvelopeHandler integrity checks with EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM  1- A MO short message control by SIM envelope is sent to the (U)SIM  2- EnvelopeHandler.getTheHandler() method is called  3- Copy the content of the EnvelopeHandler in buffer1 using EnvelopeHandler.copy()  The EnvelopeHandler.findTLV() method is called with TAG\_ADDRESS  4- A proactive command DISPLAY TEXT is sent  5- Envelope call control by SIM is sent to the (U)SIM  EnvelopeHandler.getTheHandler() method is called  6- Checked that EnvelopeHandler content is the same as in envelope call control using EnvelopeHandler.copy() and Util.arrayCompare() methods  The EnvelopeHandler.findTLV() method is called with TAG\_DEVICE\_IDENTITIES  Call Control execution is finished.  Check that the TAG\_ADDRESS is the TLV selected  7- The content of EnvelopeHandler is compared with buffer1 using Util.arrayCompare() | 1- Applet is triggered  2- No exception is thrown.  3- No exception is thrown.  5- Applet is triggered  6- No exception is thrown and the handler contains the envelope call control by SIM  7- The contents of the EnvelopeHandler shall be the same as stored in buffer1. | 4- 91 XX  Proactive command Display Text is fetched  The terminal Response of DISPLAY TEXT is sent to the (U)SIM |
| 7 | EnvelopeHandler integrity checks with EVENT\_FORMATTED\_SMS\_CB  1- A formatted cell broadcast envelope is sent to the (U)SIM  2- EnvelopeHandler.getTheHandler() method is called  3- Copy the content of the EnvelopeHandler in buffer1 using EnvelopeHandler.copy()  The EnvelopeHandler.findTLV() method is called with TAG\_CELLBROADCAST\_PAGE  4- A proactive command DISPLAY TEXT is sent  5- Envelope call control by SIM is sent to the (U)SIM  EnvelopeHandler.getTheHandler() method is called  6- Checked that EnvelopeHandler content is the same as in envelope call control using EnvelopeHandler.copy() and Util.arrayCompare() methods  The EnvelopeHandler.findTLV() method is called with TAG\_DEVICE\_IDENTITIES  Call Control execution is finished.  Check that the TAG\_CELLBROADCAST\_PAGE is the TLV selected  7- The content of EnvelopeHandler is compared with buffer1 using Util.arrayCompare() | 1- Applet is triggered  2- No exception is thrown.  3- No exception is thrown.  5- Applet is triggered  6- No exception is thrown and the handler contains the envelope call control by SIM  7- The contents of the EnvelopeHandler shall be the same as stored in buffer1. | 4- 91 XX  Proactive command Display Text is fetched  The terminal Response of DISPLAY TEXT is sent to the (U)SIM |
| 8 | EnvelopeHandler integrity checks with EVENT\_FORMATTED\_USSD  1- A formatted USSD envelope is sent to the (U)SIM  2- EnvelopeHandler.getTheHandler() method is called  3- Copy the content of the EnvelopeHandler in buffer1 using EnvelopeHandler.copy()  The EnvelopeHandler.findTLV() method is called with TAG\_USSD\_STRING  4- A proactive command DISPLAY TEXT is sent  5- Envelope call control by SIM is sent to the (U)SIM  EnvelopeHandler.getTheHandler() method is called  6- Check that EnvelopeHandler content is the same as in the envelope call control using EnvelopeHandler.copy() and Util.arrayCompare() methods  The EnvelopeHandler.findTLV() method is called with TAG\_DEVICE\_IDENTITIES  Call Control execution is finished.  Check that the TAG\_USSD\_STRING is the TLV selected  7- The content of EnvelopeHandler is compared with buffer1 using Util.arrayCompare() | 1- Applet is triggered  2- No exception is thrown.  3- No exception is thrown.  5- Applet is triggered  6- No exception is thrown and the handler contains the envelope call control by SIM  7- The contents of the EnvelopeHandler shall be the same as stored in buffer1 | 4- 91 XX  Proactive command Display Text is fetched  The terminal Response of DISPLAY TEXT is sent to the (U)SIM |
| 9 | EnvelopeHandler integrity checks with EVENT\_UNFORMATTED\_USSD  1- A unformatted USSD envelope is sent to the (U)SIM  2- EnvelopeHandler.getTheHandler() method is called  3- Copy the content of the EnvelopeHandler in buffer1 using EnvelopeHandler.copy()  The EnvelopeHandler.findTLV method is called with TAG\_DEVICE\_IDENTITIES  4- A proactive command DISPLAY TEXT is sent  5- Envelope call control by SIM is sent to the (U)SIM  EnvelopeHandler.getTheHandler() method is called  6- Check that EnvelopeHandler content is the same as in the envelope call control using EnvelopeHandler.copy() and Util.arrayCompare() methods  The EnvelopeHandler.findTLV() method is called with TAG\_DEVICE\_IDENTITIES  Call Control execution is finished.  Check that the TAG\_DEVICE\_IDENTITIES is the TLV selected  7- The content of EnvelopeHandler is compared with buffer1 using Util.arrayCompare() | 1- Applet is triggered  2- No exception is thrown.  3- No exception is thrown.  5- Applet is triggered  6- No exception is thrown and the handler contains the envelope call control by SIM  7- The contents of the EnvelopeHandler shall be the same as stored in buffer1. | 4- 91 XX  Proactive command Display Text is fetched  The terminal Response of DISPLAY TEXT is sent to the (U)SIM |
| 10 | EnvelopeHandler integrity checks with EVENT\_DOWNLOAD\_IWLAN\_ACCESS\_STATUS  1- An Download Iwlan Access Status envelope is sent to the (U)SIM  2- EnvelopeHandler.getTheHandler() method is called  3- Copy the content of the EnvelopeHandler in buffer1 using EnvelopeHandler.copy()  The EnvelopeHandler.findTLV() method is called with TAG\_WLAN\_ACCESS\_STATUS (0x4B)  4- A proactive command DISPLAY TEXT is sent  5- Envelope call control by SIM is sent to the (U)SIM  EnvelopeHandler.getTheHandler() method is called  6- Checked that EnvelopeHandler content is the same as in envelope call control using EnvelopeHandler.copy() and Util.arrayCompare() methods  The EnvelopeHandler.findTLV() method is called with TAG\_DEVICE\_IDENTITIES  Call Control execution is finished.  Check that the TAG\_WLAN\_ACCESS\_STATUS is the TLV selected  7- The content of EnvelopeHandler is compared with buffer1 using Util.arrayCompare() | 1- Applet is triggered  2- No exception is thrown.  3- No exception is thrown.  5- Applet is triggered  6- No exception is thrown and the handler contains the envelope call control by SIM  7- The contents of the EnvelopeHandler shall be the same as stored in buffer1. | 4- 91 XX  Proactive command Display Text is fetched  The terminal Response of DISPLAY TEXT is sent to the (U)SIM |

#### 5.3.2.3 USATEnvelopeHandler

Test Area Reference: Ufw\_Hin\_Uehd

5.3.2.3.1 Conformance requirements

5.3.2.3.1.1 Normal execution

- CRRN1: When available, the *USATEnvelopeHandler* shall remain available and its content shall remain unchanged from the invocation to the termination of the *processToolkit()* method.

- CRRN2: The *USATEnvelopeHandler* TLV list is filled with the Comprehension TLV data objects of the ENVELOPE APDU command. The Comprehension TLV data objects shall be provided in the order given in the ENVELOPE command data.

- CRRN3: The (U)SAT Framework shall convert the UPDATE RECORD EFSMS APDU into a COMPREHENSION TLV List containing Device Identities TLV, Address TLV, SMS TPDU TLV and AID TLV (only if the EFSMS file updated is under an ADF).

- CRRN4: The *getEnvelopeTag()* method shall return BTAG\_SMS\_PP\_DOWNLOAD.

- CRRN5: The *getLength()* method shall return the Comprehension TLV list length.

- CRRN6: The Device Identity Simple TLV is used to store the information about the absolute record number in the EFSMS file and the value of the EFSMS record status byte.

5.3.2.3.1.2 Parameter errors

No requirements.

5.3.2.3.1.3 Context Errors

No requirements.

5.3.2.3.2 Test area files

Test Source: Test\_Ufw\_Hin\_Uehd.java

Test Applet: Ufw\_Hin\_Uehd\_1.java

Cap File: Ufw\_Hin\_Uehd.cap

5.3.2.3.3 Test coverage

| CRR Number | Test Case Number |
| --- | --- |
| N1 | 1 to 7, 14, 15, 16 |
| N2 | 1 to 7, 14, 15, 16 |
| N3 | 8, 11 |
| N4 | 10, 13 |
| N5 | 9, 12 |
| N6 | 8, 11 |

5.3.2.3.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
| 1 | USATEnvelopeHandler integrity checks with EVENT\_FORMATTED\_SMS\_PP\_ENV  1- A formatted SMS PP envelope is sent to the (U)SIM  2- USATEnvelopeHandler.getTheHandler() method is called  3- Copy the content of the EnvelopeHandler in buffer1 using USATEnvelopeHandler.copy()  The USATEnvelopeHandler.findTLV() method is called with TAG\_SMS\_TPDU  4- A proactive command DISPLAY TEXT is sent  5- Envelope call control by SIM is sent to the (U)SIM  USATEnvelopeHandler.getTheHandler() method is called  6- Check that EnvelopeHandler content is the same as in the envelope call control using USATEnvelopeHandler.copy() and Util.arrayCompare() methods  The USATEnvelopeHandler.findTLV() method is called with TAG\_DEVICE\_IDENTITIES  Call Control execution is finished.  Check that the TAG\_SMS\_TPDU is the TLV selected  7- The content of USATEnvelopeHandler is compared with buffer1 using Util.arrayCompare() | 1- Applet is triggered  2- No exception is thrown.  3- No exception is thrown.  5- Applet is triggered  6- No exception is thrown and the handler contains the envelope call control by SIM  7- The contents of the EnvelopeHandler shall be the same as stored in buffer1 | 4- 91 XX  Proactive command Display Text is fetched  The terminal Response of DISPLAY TEXT is sent to the (U)SIM |
| 2 | USATEnvelopeHandler integrity checks with EVENT\_FORMATTED\_SMS\_PP\_UPD  1- Update Record EFSMS instruction single and formatted is sent to the (U)SIM  2- USATEnvelopeHandler.getTheHandler() method is called  3- Copy the content of the EnvelopeHandler in buffer1 using USATEnvelopeHandler.copy()  The USATEnvelopeHandler.findTLV() method is called with TAG\_SMS\_TPDU  4- A proactive command DISPLAY TEXT is sent  5- Envelope call control by SIM is sent to the (U)SIM  USATEnvelopeHandler.getTheHandler() method is called  6- Checked that EnvelopeHandler content is the same as in envelope call control using USATEnvelopeHandler.copy() and Util.arrayCompare() methods  The USATEnvelopeHandler.findTLV() method is called with TAG\_SMS\_TPDU  Call Control execution is finished.  Checked that the TAG\_SMS\_TPDU is the TLV selected  7- The content of USATEnvelopeHandler is compared with buffer1 using Util.arrayCompare() | 1- Applet is triggered  2- No exception is thrown.  3- No exception is thrown.  5- Applet is triggered  6- No exception is thrown and the handler contains the envelope call control by SIM  7- The contents of the EnvelopeHandler shall be the same as stored in buffer1 | 4- 91 XX  Proactive command Display Text is fetched  The terminal Response of DISPLAY TEXT is sent to the (U)SIM |
| 3 | USATEnvelopeHandler integrity checks with EVENT\_UNFORMATTED\_SMS\_PP\_ENV  1- A unformatted SMS PP envelope is sent to the (U)SIM  2- USATEnvelopeHandler.getTheHandler() method is called  3- Copy the content of the EnvelopeHandler in buffer1 using USATEnvelopeHandler.copy()  The USATEnvelopeHandler.findTLV method is called with TAG\_DEVICE\_IDENTITIES  4- A proactive command DISPLAY TEXT is sent  5- Envelope call control by SIM is sent to the (U)SIM  USATEnvelopeHandler.getTheHandler() method is called  6- Check that EnvelopeHandler content is the same as in the envelope call control using USATEnvelopeHandler.copy() and Util.arrayCompare() methods  The USATEnvelopeHandler.findTLV() method is called with TAG\_DEVICE\_IDENTITIES  Call Control execution is finished.  Check that the TAG\_DEVICE\_IDENTITIES is the TLV selected  7- The content of USATEnvelopeHandler is compared with buffer1 using Util.arrayCompare() | 1- Applet is triggered  2- No exception is thrown.  3- No exception is thrown.  5- Applet is triggered  6- No exception is thrown and the handler contains the envelope call control by SIM  7- The contents of the EnvelopeHandler shall be the same as stored in buffer1. | 4- 91 XX  Proactive command Display Text is fetched  The terminal Response of DISPLAY TEXT is sent to the (U)SIM |
| 4 | USATEnvelopeHandler integrity checks with EVENT\_UNFORMATTED\_SMS\_PP\_UPD  1- Update Record EFSMS instruction single and unformatted is sent to the (U)SIM  2- USATEnvelopeHandler.getTheHandler() method is called  3- Copy the content of the EnvelopeHandler in buffer1 using USATEnvelopeHandler.copy()  The USATEnvelopeHandler.findTLV method is called with TAG\_SMS\_TPDU  4- A proactive command DISPLAY TEXT is sent  5- Envelope call control by SIM is sent to the (U)SIM  USATEnvelopeHandler.getTheHandler() method is called  6- Check that EnvelopeHandler content is the same as in envelope call control using USATEnvelopeHandler.copy() and Util.arrayCompare() methods  The USATEnvelopeHandler.findTLV() method is called with TAG\_DEVICE\_IDENTITIES  Call Control execution is finished.  Check that the TAG\_DEVICE\_IDENTITIES is the TLV selected  7- The content of USATEnvelopeHandler is compared with buffer1 using Util.arrayCompare() | 1- Applet is triggered  2- No exception is thrown.  3- No exception is thrown.  5- Applet is triggered  6- No exception is thrown and the handler contains the envelope call control by SIM  7- The contents of the EnvelopeHandler shall be the same as stored in buffer1. | 4- 91 XX  Proactive command Display Text is fetched  The terminal Response of DISPLAY TEXT is sent to the (U)SIM |
| 5 | USATEnvelopeHandler integrity checks with EVENT\_UNFORMATTED\_SMS\_CB  1- An unformatted cell broadcast envelope is sent to the (U)SIM  2- USATEnvelopeHandler.getTheHandler() method is called  3- Copy the content of the EnvelopeHandler in buffer1 using USATEnvelopeHandler.copy()  The USATEnvelopeHandler.findTLV() method is called with TAG\_CELLBROADCAST\_PAGE  4- A proactive command DISPLAY TEXT is sent  5- Envelope call control by SIM is sent to the (U)SIM  USATEnvelopeHandler.getTheHandler() method is called  6- Checked that EnvelopeHandler content is the same as in envelope call control using USATEnvelopeHandler.copy() and Util.arrayCompare() methods  The USATEnvelopeHandler.findTLV() method is called with TAG\_DEVICE\_IDENTITIES  Call Control execution is finished.  Check that the TAG\_CELLBROADCAST\_PAGE is the TLV selected  7- The content of USATEnvelopeHandler is compared with buffer1 using Util.arrayCompare() | 1- Applet is triggered  2- No exception is thrown.  3- No exception is thrown.  5- Applet is triggered  6- No exception is thrown and the handler contains the envelope call control by SIM  7- The contents of the EnvelopeHandler shall be the same as stored in buffer1. | 4- 91 XX  Proactive command Display Text is fetched  The terminal Response of DISPLAY TEXT is sent to the (U)SIM |
| 6 | USATEnvelopeHandler integrity checks with EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM  1- A MO short message control by SIM envelope is sent to the (U)SIM  2- USATEnvelopeHandler.getTheHandler() method is called  3- Copy the content of the EnvelopeHandler in buffer1 using USATEnvelopeHandler.copy()  The USATEnvelopeHandler.findTLV() method is called with TAG\_ADDRESS  4- A proactive command DISPLAY TEXT is sent  5- Envelope call control by SIM is sent to the (U)SIM  USATEnvelopeHandler.getTheHandler() method is called  6- Checked that EnvelopeHandler content is the same as in envelope call control using USATEnvelopeHandler.copy() and Util.arrayCompare() methods  The USATEnvelopeHandler.findTLV() method is called with TAG\_DEVICE\_IDENTITIES  Call Control execution is finished.  Check that the TAG\_ADDRESS is the TLV selected  7- The content of USATEnvelopeHandler is compared with buffer1 using Util.arrayCompare() | 1- Applet is triggered  2- No exception is thrown.  3- No exception is thrown.  5- Applet is triggered  6- No exception is thrown and the handler contains the envelope call control by SIM  7- The contents of the EnvelopeHandler shall be the same as stored in buffer1. | 4- 91 XX  Proactive command Display Text is fetched  The terminal Response of DISPLAY TEXT is sent to the (U)SIM |
| 7 | USATEnvelopeHandler integrity checks with EVENT\_FORMATTED\_SMS\_CB  1- A formatted cell broadcast envelope is sent to the (U)SIM  2- USATEnvelopeHandler.getTheHandler() method is called  3- Copy the content of the EnvelopeHandler in buffer1 using USATEnvelopeHandler.copy()  The USATEnvelopeHandler.findTLV() method is called with TAG\_CELLBROADCAST\_PAGE  4- A proactive command DISPLAY TEXT is sent  5- Envelope call control by SIM is sent to the (U)SIM  USATEnvelopeHandler.getTheHandler() method is called  6- Checked that EnvelopeHandler content is the same as in envelope call control using USATEnvelopeHandler.copy() and Util.arrayCompare() methods  The USATEnvelopeHandler.findTLV() method is called with TAG\_DEVICE\_IDENTITIES  Call Control execution is finished.  Check that the TAG\_CELLBROADCAST\_PAGE is the TLV selected  7- The content of USATEnvelopeHandler is compared with buffer1 using Util.arrayCompare() | 1- Applet is triggered  2- No exception is thrown.  3- No exception is thrown.  5- Applet is triggered  6- No exception is thrown and the handler contains the envelope call control by SIM  7- The contents of the EnvelopeHandler shall be the same as stored in buffer1. | 4- 91 XX  Proactive command Display Text is fetched  The terminal Response of DISPLAY TEXT is sent to the (U)SIM |
| 8 | Check the TLV list conversion for EVENT\_FORMATTED\_SMS\_PP\_UPD  1- An EVENT\_FORMATTED\_SMS\_PP\_UPD is sent to the (U)SIM.  2- The findTLV(tag == device identities Tag) method is called.  3- The getValueByte(offset == 0) method is called.  4- The getValueByte(offset == 1) method is called.  5- The findTLV(tag == address Tag) method is called.  6- Check the content.  7- The findTLV(tag == SMS TPDU Tag) method is called.  8- Check the content. | 1- Applet is triggered.  2- No exception is thrown.  3- return the absolute record.  4- return the record status.  5- No exception is thrown.  7- No exception is thrown. |  |
| 9 | getLength() call  Call getLength() method | return the Comprehension TLV list length |  |
| 10 | getEnvelopeTag() call  Call getTag() method | return BTAG\_SMS\_PP\_DOWNLOAD |  |
| 11 | Check the TLV list conversion for EVENT\_UNFORMATTED\_SMS\_PP\_UPD  1- An EVENT\_UNFORMATTED\_SMS\_PP\_UPD is sent to the (U)SIM.  2- The findTLV(tag == device identities Tag) method is called.  3- The getValueByte(offset == 0) method is called.  4- The getValueByte(offset == 1) method is called.  5- The findTLV(tag == address Tag) method is called.  6- Check the content.  7- The findTLV(tag == SMS TPDU Tag) method is called.  8- Check the content. | 1- Applet is triggered.  2- No exception is thrown.  3- return the absolute record.  4- return the record status.  5- No exception is thrown.  7- No exception is thrown. |  |
| 12 | getLength() call  Call getLength() method | return the Comprehension TLV list length |  |
| 13 | getEnvelopeTag() call  Call getEnvelopeTag() method | return BTAG\_SMS\_PP\_DOWNLOAD |  |
| 14 | USATEnvelopeHandler integrity checks with EVENT\_FORMATTED\_USSD  1- A formatted USSD envelope is sent to the (U)SIM  2- USATEnvelopeHandler.getTheHandler() method is called  3- Copy the content of the EnvelopeHandler in buffer1 using USATEnvelopeHandler.copy()  The USATEnvelopeHandler.findTLV() method is called with TAG\_USSD\_STRING  4- A proactive command DISPLAY TEXT is sent  5- Envelope call control by SIM is sent to the (U)SIM  USATEnvelopeHandler.getTheHandler() method is called  6- Check that EnvelopeHandler content is the same as in the envelope call control using USATEnvelopeHandler.copy() and Util.arrayCompare() methods  The USATEnvelopeHandler.findTLV() method is called with TAG\_DEVICE\_IDENTITIES  Call Control execution is finished.  Check that the TAG\_USSD\_STRING is the TLV selected  7- The content of USATEnvelopeHandler is compared with buffer1 using Util.arrayCompare() | 1- Applet is triggered  2- No exception is thrown.  3- No exception is thrown.  5- Applet is triggered  6- No exception is thrown and the handler contains the envelope call control by SIM  7- The contents of the EnvelopeHandler shall be the same as stored in buffer1 | 4- 91 XX  Proactive command Display Text is fetched  The terminal Response of DISPLAY TEXT is sent to the (U)SIM |
| 15 | USATEnvelopeHandler integrity checks with EVENT\_UNFORMATTED\_USSD  1- A unformatted USSD envelope is sent to the (U)SIM  2- USATEnvelopeHandler.getTheHandler() method is called  3- Copy the content of the EnvelopeHandler in buffer1 using USATEnvelopeHandler.copy()  The USATEnvelopeHandler.findTLV method is called with TAG\_DEVICE\_IDENTITIES  4- A proactive command DISPLAY TEXT is sent  5- Envelope call control by SIM is sent to the (U)SIM  USATEnvelopeHandler.getTheHandler() method is called  6- Check that EnvelopeHandler content is the same as in the envelope call control using USATEnvelopeHandler.copy() and Util.arrayCompare() methods  The USATEnvelopeHandler.findTLV() method is called with TAG\_DEVICE\_IDENTITIES  Call Control execution is finished.  Check that the TAG\_DEVICE\_IDENTITIES is the TLV selected  7- The content of USATEnvelopeHandler is compared with buffer1 using Util.arrayCompare() | 1- Applet is triggered  2- No exception is thrown.  3- No exception is thrown.  5- Applet is triggered  6- No exception is thrown and the handler contains the envelope call control by SIM  7- The contents of the EnvelopeHandler shall be the same as stored in buffer1. | 4- 91 XX  Proactive command Display Text is fetched  The terminal Response of DISPLAY TEXT is sent to the (U)SIM |
| 16 | USATEnvelopeHandler integrity checks with EVENT\_DOWNLOAD\_IWLAN\_ACCESS\_STATUS  1- A Download Iwlan Access Status envelope is sent to the (U)SIM  2- USATEnvelopeHandler.getTheHandler() method is called  3- Copy the content of the EnvelopeHandler in buffer1 using USATEnvelopeHandler.copy()  The USATEnvelopeHandler.findTLV method is called with TAG\_DEVICE\_IDENTITIES  4- A proactive command DISPLAY TEXT is sent  5- Envelope call control by SIM is sent to the (U)SIM  USATEnvelopeHandler.getTheHandler() method is called  6- Check that EnvelopeHandler content is the same as in the envelope call control using USATEnvelopeHandler.copy() and Util.arrayCompare() methods  The USATEnvelopeHandler.findTLV() method is called with TAG\_DEVICE\_IDENTITIES  Call Control execution is finished.  Check that the TAG\_DEVICE\_IDENTITIES is the TLV selected  7- The content of USATEnvelopeHandler is compared with buffer1 using Util.arrayCompare() | 1- Applet is triggered  2- No exception is thrown.  3- No exception is thrown.  5- Applet is triggered  6- No exception is thrown and the handler contains the envelope call control by SIM  7- The contents of the EnvelopeHandler shall be the same as stored in buffer1. | 4- 91 XX  Proactive command Display Text is fetched  The terminal Response of DISPLAY TEXT is sent to the (U)SIM |

### 5.3.3 Exception handling

#### 5.3.3.1 General Behaviour

Test Area Reference: Ufw\_Exh\_Genb

5.3.3.1.1 Conformance requirement

5.3.3.1.1.1 Normal execution

- CRRN1: If more than one Applet shall be triggered by the currently processed event all Exceptions shall be caught by the USAT Framework and shall not be sent to the terminal. The USAT Framework shall proceed with the triggering.

- CRRN2: If only one Applet shall be triggered by the currently processed event and an ISOException with the reason code REPLY\_BUSY is thrown, it shall be sent to the terminal using the Status Word 0x9300.

- CRRN3: If only one Applet shall be triggered by the currently processed event other Exceptions than an ISOException with the reason code REPLY\_BUSY shall not be propagated to the terminal.

5.3.3.1.1.2 Parameter errors

No requirements.

5.3.3.1.1.3 Context errors

No requirements.

5.3.3.1.2 Test area files

Test Source: Test\_Ufw\_Exh\_Genb.java

Test Applet: Ufw\_Exh\_Genb\_1.java

Ufw\_Exh\_Genb\_2.java

Cap File: ufw\_exh\_genb.cap

5.3.3.1.3 Test coverage

| CRR Number | Test Case Number |
| --- | --- |
| CRRN1 | 1, 2, 3, 4 |

5.3.3.1.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
| 0 | Applet1 is installed and registers to EVENT\_FORMATTED\_SMS\_PP\_ENV EVENT\_UNFORMATTED\_SMS\_PP\_ENV  EVENT\_FORMATTED\_USSD  and  Applet2 is installed and registers to  EVENT\_UNFORMATTED\_SMS\_PP\_ENV and EVENT\_UNFORMATTED\_SMS\_CB  EVENT\_UNFORMATTED\_USSD |  |  |
| 1 | ISOException REPLY\_BUSY is not sent to the terminal in multi triggering  1- Send an envelope Event Unformatted SMS PP ENV (multi triggering event, multi registered applets)  4- Send an envelope Event Unformatted SMS PP ENV (multi triggering event, multi registered applets) | 1- Applet1 is triggered  2- Applet1 sends a ISOException with the reason code REPLY\_BUSY then finalizes  Applet2 is triggered, does nothing and finalizes  4- Applet1 is triggered, does nothing and finalizes  Applet2 is triggered, sends a ISOException with the reason code REPLY\_BUSY then finalizes | 3- SW = 90 00  5- SW = 90 00 |
| 2 | ISOException REPLY\_BUSY is sent to the terminal in single triggering SMS PP  1- Send an envelope Event formatted SMS PP ENV to trigger Applet1 (single triggering event)  2- Send an envelope Event Unformatted SMS CB (multi triggering event, single registered applet) | 1- Applet1 is triggered, sends a ISOException with the reason code REPLY\_BUSY then finalizes  2- Applet2 is triggered, sends a ISOException with the reason code REPLY\_BUSY then finalizes | 1- SW = 93 00  2- SW = 93 00 |
| 3 | Other exception than ISOException REPLY\_BUSY are not sent to the terminal  1- Send an envelope Event formatted SMS PP ENV to trigger Applet1 (single triggering event)  2- Send an envelope Event formatted SMS PP ENV to trigger Applet1 (single triggering event) | 1- Applet1 is triggered, sends a ISOException with reason code different to REPLY\_BUSY then finalizes  2- Applet1 is triggered, sends a ToolkitException then finalizes | 1- SW = 90 00  2- SW = 90 00 |
| 4 | ISOException REPLY\_BUSY is sent to the terminal in single triggering USSD  1- Send an envelope Event formatted USSD to trigger Applet1 (single triggering event)  2- Send an envelope Event unformatted SMS CB (multi triggering event, single registered applet) | 1- Applet1 is triggered, sends a ISOException with the reason code REPLY\_BUSY then finalizes  2- Applet2 is triggered, sends a ISOException with the reason code REPLY\_BUSY then finalizes | 1- SW = 93 00  2- SW = 93 00 |

#### 5.3.3.2 Interaction with Multiple Triggering

Test Area Reference: Ufw\_Exh\_Imtg

5.3.3.2.1 Conformance requirement

5.3.3.2.1.1 Normal execution:

- CRRN1: An exception thrown by a toolkit applet, will not influence toolkit applets registered to the same event.

5.3.3.2.1.2 Parameter errors

No requirements.

5.3.3.2.1.3 Context errors

No requirements.

5.3.3.2.2 Test area files

Test Source: Test\_Ufw\_Exh\_Imtg.java

Test Applet: Ufw\_Exh\_Imtg\_1.java

Ufw\_Exh\_Imtg\_2.java

Cap File: ufw\_exh\_imtg.cap

5.3.3.2.3 Test coverage

| CRR Number | Test Case Number |
| --- | --- |
| CRRN1 | 1, 2, 3, 4 |

5.3.3.2.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
| 0 | Load/install 2 toolkit applets registered to EVENT\_UNFORMATTED\_SMS\_PP\_ENV, EVENT\_UNFORMATTED\_SMS\_PP\_UPD, EVENT\_UNFORMATTED\_SMS\_CB,  EVENT\_UNFORMATTED\_USSD  Applet1: Priority= 0x01,  Applet2: Priority= 0x02,  (i.e. Applet1 is triggered before Applet2) |  |  |
| 1 | UNFORMATTED\_SMS\_PP\_ENV is sent | 1- Applet1 is triggered    2- NullPointerException is thrown  3- Applet2 is triggered |  |
| 2 | UNFORMATTED\_SMS\_PP\_UPD is sent | 1- Applet1 is triggered    2- NullPointerException is thrown  3- Applet2 is triggered |  |
| 3 | UNFORMATTED\_SMS\_CB is sent | 1- Applet1 is triggered    2- NullPointerException is thrown  3- Applet2 is triggered |  |
| 4 | UNFORMATTED\_USSD is send | 1- Applet1 is triggered    2- NullPointerException is thrown  3- Applet2 is triggered |  |

### 5.3.4 Applet triggering

#### 5.3.4.1 EVENT\_FORMATTED\_SMS\_PP\_ENV

Test Area Reference: Ufw\_Apt\_Efse

5.3.4.1.1 Conformance requirement

5.3.4.1.1.1 Normal execution

- CRRN1: The applet is triggered by the EVENT\_FORMATTED\_SMS\_PP\_ENV once:

- it has been registered to this event;

- a Short Message Point to Point (Single or Concatenated) is received by Envelope APDU(s) and is formatted according to TS 31.115 [10];

- the toolkit applet to be triggered is registered with the corresponding TAR in the SMS TPDU;

- the security is verified.

- CRRN2: The applet is not triggered by the EVENT\_FORMATTED\_SMS\_PP\_ENV once it has deregistered from this event.

5.3.4.1.1.2 Parameters error

No requirements.

5.3.4.1.1.3 Context Errors

No requirements.

5.3.4.1.2 Test area files

Test Source: Test\_Ufw\_Apt\_Efse.java

Test Applet: Ufw\_Apt\_Efse\_1.java

Cap File: ufw\_apt\_efse.cap

5.3.4.1.3 Test coverage

| CRR Number | Test Case Number |
| --- | --- |
| CRRN1 (See note) | 1, 2 |
| CRRN2 | 2 |

NOTE: The security checks are not relevant to the test designed in this test area; they will be checked in the "Framework Security Management" clause.

5.3.4.1.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
| 1 | Applet registration to EVENT FORMATTED\_SMS\_PP\_ENV and triggering  Applet is registered to EVENT\_FORMATTED\_SMS\_PP\_ENV and EVENT\_UNRECOGNIZED\_ENVELOPE  1- A Single Short Message SMS-PP Formatted Data Download is sent to the USIM.  2- A Concatenated Short Message SMS-PP Formatted Data Download is sent to the USIM (composed of 2 Short Messages. The UDL for the first Short Message is 70 and for the second 70) | 1- Applet is triggered  2- Applet is triggered |  |
| 2 | Applet deregistration  ToolkitRegistry.clearEvent() method is called for EVENT\_FORMATTED\_SMS\_PP\_ENV  1- A Single Short Message SMS-PP Data Download is sent to the USIM.  2- A Concatenated Short Messages SMS-PP Data Download is sent to the USIM (composed of 2 Short Messages. The UDL for the first Short Message is 70 and for the second 70).  An unrecognized envelope is sent to the USIM  ToolkitRegistry.setEvent() method is called for EVENT\_FORMATTED\_SMS\_PP\_ENV  3- A Single Short Messages SMS-PP Data Download is sent to the USIM.  4- A Concatenated Short Messages SMS-PP Data Download is sent to the USIM (composed of 2 Short Messages. The UDL for the first Short Message is 70 and for the second 70). | 1- Applet is not triggered  2- Applet is not triggered  3- Applet is triggered  4- Applet is triggered |  |

#### 5.3.4.2 EVENT\_UNFORMATTED\_SMS\_PP\_ENV

Test Area Reference: Ufw\_Apt\_Euse

5.3.4.2.1 Conformance requirement

5.3.4.2.1.1 Normal execution

- CRRN1: The applets registers are triggered by the EVENT\_UNFORMATTED\_SMS\_PP\_ENV once a Short Message Point to Point (Single or Concatenated) is received by Envelope APDU(s) and is unformatted.

- CRRN2: The applet is not triggered by the EVENT\_UNFORMATTED\_SMS\_PP\_ENV once it has deregistered from this event.

5.3.4.2.1.2 Parameters error

No requirements.

5.3.4.2.1.3 Context Errors

No requirements.

5.3.4.2.2 Test area files

Test Source: Test\_Ufw\_Apt\_Euse.java

Test Applet: Ufw\_Apt\_Euse\_1.java

Cap File: ufw\_apt\_euse.cap

5.3.4.2.3 Test coverage

| CRR Number | Test Case Number |
| --- | --- |
| CRRN1 | 1, 2 |
| CRRN2 | 2 |

5.3.4.2.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
| 1 | Applet registration to EVENT\_UNFORMATTED\_SMS\_PP\_ENV and triggering  Applet is registered to the EVENT\_UNFORMATTED\_SMS\_PP\_ENV and EVENT\_FORMATTED\_SMS\_PP\_ENV.  1-ToolkitRegistry.isEventSet() method is called for EVENT\_UNFORMATTED\_SMS\_PP\_ENV  2- A Single and Unformatted SMS-PP Data Download Envelope is sent to the USIM.  3- A Concatenated and Unformatted SMS-PP Data Download Envelope is sent to the USIM (composed of 2 Short Messages. The UDL for the first Short Message is 70 and for the second 70) | 1- The method returns true  2- Applet is triggered  3- Applet is triggered |  |
| 2 | Applet deregistration  ToolkitRegistry.clearEvent()method is called for EVENT\_UNFORMATTED\_SMS\_PP\_ENV  1- A Single and Unformatted SMS-PP Data Download Envelope is sent to the USIM.  2- A Concatenated and Unformatted SMS-PP Data Download Envelope is sent to the USIM (composed of 2 Short Messages. The UDL for the first Short Message is 70 and for the second 70)  An unrecognized envelope is sent to the USIM  ToolkitRegistry.setEvent() method is called for EVENT\_UNFORMATTED\_SMS\_PP\_ENV  3- A Single and Unformatted SMS-PP Data Download Envelope is sent to the USIM.  4- A Concatenated and Unformatted SMS-PP Data Download Envelope is sent to the USIM (composed of 2 Short Messages. The UDL for the first Short Message is 70 and for the second 70) | 1- Applet isn't triggered  2- Applet isn't triggered  3- Applet is triggered  4- Applet is triggered |  |

#### 5.3.4.3 EVENT\_FORMATTED\_SMS\_PP\_UPD

Test Area Reference: Ufw\_Apt\_Efsu

5.3.4.3.1 Conformance requirement

5.3.4.3.1.1 Normal execution

- CRRN1: The applet is triggered by the EVENT\_FORMATTED\_SMS\_PP\_UPD once:

- it has been registered to this event,

- a Short Message Point to Point (Single or Concatenated) is received by Update Record EFSMS APDU(s) and is formatted according to TS 31.115[10],

- the toolkit applet to be triggered is registered with the corresponding TAR in the SMS TPDU,

- CRRN2: The applets are not triggered by the EVENT\_FORMATTED\_SMS\_PP\_UPD once it has deregistered from this event.

5.3.4.3.2 Test area files

Test Source: Test\_Ufw\_Apt\_Efsu.java

Test Applet: Ufw\_Apt\_Efsu\_1.java

Cap File: ufw\_apt\_efsu.cap

5.3.4.3.3 Test coverage

| CRR Number | Test Case Number |
| --- | --- |
| CRRN1 (See note) | 1,2 |
| CRRN2 | 2 |

NOTE: The security checks are not relevant to the test designed in this test area; they will be checked in the "Framework Security Management" clause.

5.3.4.3.4 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| 1 | Applet registration to EVENT FORMATTED\_SMS\_PP\_UPD and triggering  Applet is registered to EVENT\_FORMATTED\_SMS\_PP\_UPD and EVENT\_UNRECOGNIZED\_ENVELOPE  1. ToolkitRegistry.isEventSet() method is called for EVENT\_FORMATTED\_SMS\_PP\_UPD  2. Short Message Point to Point Single and Formatted is received by Update Record EFSMS APDU.  3. Short Message Point to Point Concatenated Formatted is received by Update Record EFSMS APDU(s) (The Concatenated Message is composed of 2 Short Messages. The UDL for the first Short Message is 70 and for the second 70). | 1- The method returns true.  2- Applet is triggered.  3- Applet is triggered on reception of the last concatenated SMS |  |
| 2 | Applet deregistration  ToolkitRegistry.clearEvent() method is called for EVENT\_FORMATTED\_SMS\_PP\_UPD  1. Short Message Point to Point Single and Formatted is received by Update Record EFSMS APDU.  2. Short Message Point to Point Concatenated and Formatted is received by Update Record EFSMS APDU(s). (The Concatenated Message is composed of 2 Short Messages. The UDL for the first Short Message is 70 and for the second 70).  An unrecognized envelope is sent to the USIM  ToolkitRegistry.setEvent() method is called for EVENT\_FORMATTED\_SMS\_PP\_UPD  3. Short Message Point to Point Single and Formatted is received by Update Record EFSMS APDU.  4. Short Message Point to Point Concatenated Formatted is received by Update Record EFSMS APDU(s). (The Concatenated Message is composed of 2 Short Messages. The UDL for the first Short Message is 70 and for the second 70). | 1- Applet is not triggered  2- Applet is not triggered  3- Applet is triggered  4- Applet is triggered on reception of the last concatenated SMS. |  |

#### 5.3.4.4 EVENT\_UNFORMATTED\_SMS\_PP\_UPD

Test Area Reference: Ufw\_Apt\_Eusu

5.3.4.4.1 Conformance requirement

5.3.4.4.1.1 Normal execution

- CRRN1: The applets registers are triggered by the EVENT\_UNFORMATTED\_SMS\_PP\_UPD once a Short Message Point to Point (Single or Concatenated) is received by Update Record EFSMS APDU(s) and is unformatted.

- CRRN2: The applets are not triggered by the EVENT\_UNFORMATTED\_SMS\_PP\_UPD once it has deregistered from this event.

5.3.4.4.2 Test area files

Test Source: Test\_Ufw\_Apt\_Eusu.java

Test Applet: Ufw\_Apt\_Eusu\_1.java

Cap File: ufw\_apt\_eusu.cap

5.3.4.4.3 Test coverage

| CRR Number | Test Case Number |
| --- | --- |
| CRRN1 | 1,2 |
| CRRN2 | 2 |

5.3.4.4.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
| 1 | Applet registration to EVENT UNFORMATTED\_SMS\_PP\_UPD and triggering  Applet is registered to EVENT\_UNFORMATTED\_SMS\_PP\_UPD and EVENT\_UNRECOGNIZED\_ENVELOPE  1. ToolkitRegistry.isEventSet() method is called for EVENT\_UNFORMATTED\_SMS\_PP\_UPD  2. Short Message Point to Point Single and Unformatted is received by Update Record EFSMS APDU  3. Short Message Point to Point Concatenated and Unformatted is received by Update Record EFSMS APDU (The Concatenated Message is composed of 2 Short Messages. The UDL for the first Short Message is 70 and for the second 70). | 1- Applet is not triggered  2- Applet is triggered.  3- Applet is triggered on reception of the last concatenated SMS. |  |
| 2 | Applet deregistration  ToolkitRegistry.clearEvent() method is called for EVENT\_UNFORMATTED\_SMS\_PP\_UPD  1. Short Message Point to Point Single and Unformatted is received by Update Record EFSMS APDU  2. Short Message Point to Point Concatenated and Unformatted is received by Update Record EFSMS APDU(s) (The Concatenated Message is composed of 2 Short Messages. The UDL for the first Short Message is 70 and for the second 70).  An unrecognized envelope is sent to the USIM  ToolkitRegistry.setEvent() method is called for EVENT\_UNFORMATTED\_SMS\_PP\_UPD  3. Short Message Point to Point Single and Unformatted is received by Update Record EFSMS APDU  4. Short Message Point to Point Concatenated and Unformatted is received by Update Record EFSMS APDU(s) (The Concatenated Message is composed of 2 Short Messages. The UDL for the first Short Message is 70 and for the second 70). | 1- Applet is not triggered  2- Applet is not triggered.  3- Applet is triggered  4- Applet is triggered on reception of the last concatenated SMS |  |

#### 5.3.4.5 EVENT\_FORMATTED\_SMS\_CB

Test Area Reference: Ufw\_Apt\_Efcb

5.3.4.5.1 Conformance requirement

5.3.4.5.1.1 Normal execution

- CRRN1: The applet is triggered by the EVENT\_FORMATTED\_SMS\_CB once:

- it has been registered to this event;

- an envelope APDU carrying a Cell Broadcast Page, formatted according to TS 31.115[10] , is received;

- the toolkit applet to be triggered is registered with the corresponding TAR in the CB page;

- the security is verified.

- CRRN2: The applet is not triggered by the EVENT\_FORMATTED\_SMS\_CB once it has deregistered from this event.

5.3.4.5.1.2 Parameters error

No requirements.

5.3.4.5.1.3 Context Errors

No requirements.

5.3.4.5.2 Test area files

Test Source: Test\_Ufw\_Apt\_Efcb.java

Test Applet: Ufw\_Apt\_Efcb\_1.java

Cap File: ufw\_apt\_efcb.cap

5.3.4.5.3 Test coverage

| CR Number | Test Case Number |
| --- | --- |
| CRRN1 (See note) | 1, 2 |
| CRRN2 | 2 |
| NOTE: The security checks are not relevant to the test designed in this test area; they will be checked in clause 6.3.6. | |

5.3.4.5.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
| 1 | Applet registration to EVENT\_FORMATTED\_SMS\_CB and triggering  Applet is registered to EVENT\_FORMATTED\_SMS\_CB and EVENT\_FORMATTED\_SMS\_PP\_ENV  1-An Envelope EVENT\_FORMATTED\_SMS\_CB is sent to the USIM. | 1-Applet is triggered |  |
| 2 | Applet deregistration  ToolkitRegistry.clearEvent() method is called for EVENT\_FORMATTED\_SMS\_CB  1-A formatted SMS CB envelope is sent to the USIM.  2-An envelope SMS-PP formatted is sent to the USIM  ToolkitRegistry.setEvent() method is called for EVENT\_FORMATTED\_SMS\_CB  3-An Envelope FORMATTED\_SMS\_CB is sent to the USIM | 1- Applet is not triggered  2- Applet is triggered  3- Applet is triggered |  |

#### 5.3.4.6 EVENT\_UNFORMATTED\_SMS\_CB

Test Area Reference: Ufw\_Apt\_Eucb

5.3.4.6.1 Conformance requirement

5.3.4.6.1.1 Normal execution

- CRRN1: The applet is triggered by the EVENT\_UNFORMATTED\_SMS\_CB once it has registered to this event and an Envelope Cell Broadcast Download is received.

- CRRN2: The applet is not triggered by the EVENT\_UNFORMATTED\_SMS\_CB once it has deregistered from this event.

5.3.4.6.1.2 Parameters error

No requirements.

5.3.4.6.1.3 Context Errors

No requirements.

5.3.4.6.2 Test area files

Test Source: Test\_Ufw\_Apt\_Eucb.java

Test Applet: Ufw\_Apt\_Eucb\_1.java

Cap File: ufw\_apt\_eucb.cap

5.3.4.5.3 Test coverage

| CRR Number | Test Case Number |
| --- | --- |
| CRRN1 | 1, 2 |
| CRRN2 | 2 |

5.3.4.6.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
| 1 | Applet registration to EVENT\_UNFORMATTED\_SMS\_CB and triggering  Applet is registered to the EVENT\_UNFORMATTED\_SMS\_CB and EVENT\_FORMATTED\_SMS\_PP\_ENV.  event= EVENT\_UNFORMATTED\_SMS\_CB  1-ToolkitRegistry.isEventSet() method is called.  2-An Envelope UNFORMATTED\_SMS\_CB is sent to the USIM. | 1- Method returns true.  2- Applet is triggered |  |
| 2 | Applet deregistration  ToolkitRegistry.ClearEvent()method is called for EVENT\_UNFORMATTED\_SMS\_CB  1-An Envelope UNFORMATTED\_SMS\_CB is sent to the USIM.  An Envelope formatted SMS pp envelope is sent to the USIM  event= EVENT\_UNFORMATTED\_SMS\_CB  ToolkitRegistry.setEvent() method is called for EVENT\_UNFORMATTED\_SMS\_CB  2-An Envelope UNFORMATTED\_SMS\_CB is sent to the USIM. | 1- Applet isn't triggered  2- Applet is triggered |  |

#### 5.3.4.7 EVENT\_FORMATTED\_USSD

Test Area Reference: Ufw\_Apt\_Efud

5.3.4.7.1 Conformance requirement

5.3.4.7.1.1 Normal execution

- CRRN1: The applet is triggered by the EVENT\_FORMATTED\_USSD once:

- it has been registered to this event;

- formatted according to TS 31.115 [10] or an other protocol to identify explicitly the toolkit applet for which the message is sent;

- the toolkit applet to be triggered is registered with the corresponding TAR in the USSD String TLV;

- the security is verified.

- CRRN2: The applet is not triggered by the EVENT\_FORMATTED\_USSD once it has deregistered from this event.

5.3.4.7.1.2 Parameters error

No requirements.

5.3.4.7.1.3 Context Errors

No requirements.

5.3.4.7.2 Test area files

Test Source: Test\_Ufw\_Apt\_ Efud.java

Test Applet: Ufw\_Apt\_ Efud \_1.java

Cap File: ufw\_apt\_ Efud.cap

5.3.4.7.3 Test coverage

| CRR Number | Test Case Number |
| --- | --- |
| CRRN1 | 1,2 |
| CRRN2 | 2 |

NOTE: The security checks are not relevant to the test designed in this test area; they will be checked in the "Framework Security Management" clause.

5.3.4.7.4 Test coverage

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
| 1 | Applet registration to EVENT FORMATTED\_USSD and triggering  Applet is registered to EVENT\_FORMATTED\_USSD and EVENT\_UNRECOGNIZED\_ENVELOPE  1- A Single USSD Formatted Envelope is sent to the USIM.  2- A Concatenated USSD is sent to the USIM (composed of 2 USSD messages. The UDL for the first Short Message is 70 and for the second 70) | 1- Applet is triggered  2- Applet is triggered |  |
|  | Applet deregistration  ToolkitRegistry.clearEvent() method is called for EVENT\_FORMATTED\_USSD  1- A formatted USSD message is sent to the USIM.  2- A Concatenated USSD message is sent to the USIM (composed of 2 USSD Messages. The UDL for the first USSD message is 70 and for the second 70).  An unrecognized envelope is sent to the USIM  ToolkitRegistry.setEvent() method is called for EVENT\_FORMATTED\_USSD  3- A Single formatted USSD is sent to the USIM.  4- A Concatenated USSD message is sent to the USIM (composed of 2 Messages. The UDL for the first USSD message is 70 and for the second 70). | 1- Applet is not triggered  2- Applet is not triggered  3- Applet is triggered  4- Applet is triggered |  |

#### 5.3.4.8 EVENT\_UNFORMATTED\_USSD

Test Area Reference: Ufw\_Apt\_Euud

5.3.4.8.1 Conformance requirement

5.3.4.8.1.1 Normal execution

- CRRN1: The applets registers are triggered by the EVENT\_UNFORMATTED\_USSD once a unformatted USSDis received by Envelope APDU(s) and is unformatted.

- CRRN2: The applet is not triggered by the EVENT\_UNFORMATTED\_USSD once it has deregistered from this event.

5.3.4.8.1.2 Parameters error

No requirements.

5.3.4.8.1.3 Context Errors

No requirements.

5.3.4.8.2 Test area files

Test Source: Test\_Ufw\_Apt\_Euud.java

Test Applet: Ufw\_Apt\_Euud\_1.java

Cap File: ufw\_apt\_euud.cap

5.3.4.8.3 Test coverage

| CRR Number | Test Case Number |
| --- | --- |
| CRRN1 | 1,2 |
| CRRN2 | 2 |

5.3.4.8.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
| 1 | Applet registration to EVENT\_UNFORMATTED\_USSD and triggering  Applet is registered to the EVENT\_UNFORMATTED\_USSD and EVENT\_FORMATTED\_USSD  1-ToolkitRegistry.isEventSet() method is called for EVENT\_UNFORMATTED\_USSD  2- A Single and Unformatted USSD message is sent to the USIM.  3- A Concatenated and Unformatted USSD message is sent to the USIM (composed of 2 USSD messages. The UDL for the first message is 70 and for the second 70) | 1- The method returns true  2- Applet is triggered  3- Applet is triggered |  |
|  | Applet deregistration  ToolkitRegistry.clearEvent()method is called for EVENT\_UNFORMATTED\_USSD  1- A Single and Unformatted USSD message is sent to the USIM.  2- A Concatenated and Unformatted USSD message is sent to the USIM (composed of 2 USSD messages. The UDL for the first Message is 70 and for the second 70)  An unrecognized envelope is sent to the USIM  ToolkitRegistry.setEvent() method is called for EVENT\_UNFORMATTED\_USSD  3- A Single and Unformatted USSD Envelope is sent to the USIM.  4- A Concatenated and Unformatted USSD Envelope is sent to the USIM (composed of 2 messages. The UDL for the first message is 70 and for the second 70) | 1- Applet is not triggered  2- Applet is not triggered  3- Applet is triggered  4- Applet is triggered |  |

### 5.3.5 Envelope response posting

#### 5.3.5.1 EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM

Test Area Reference: Ufw\_Erp\_Emcn

5.3.5.1.1 Conformance requirement

5.3.5.1.1.1 Normal execution

- CRRN1: The (U)SAT Framework can't reply busy when an Envelope(MO-Short Message Control) is sent to the (U)SAT.

5.3.6.2.1.2 Parameter errors

No requirements.

5.3.6.2.1.3 Context errors

No requirements.

5.3.5.1.2 Test area files

Test Source: Test\_Ufw\_Erp\_Emcn.scr

Test Applet: Ufw\_Erp\_Emcn\_1.java

Ufw\_Erp\_Emcn\_2.java

Ufw\_Erp\_Emcn\_3.java

Cap File: Ufw\_Erp\_Emcn.cap

5.3.5.1.3 Test coverage

| CRR Number | Test Case Number |
| --- | --- |
| CRRN1 | 1, 2 |

5.3.5.1.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
| 1 | Applet1 is registered on the EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM; Applet2 is registered and triggered on the EVENT\_MENU\_SELECTION.  1- Applet2 invokes the method send() and no fetch is performed  2- Envelope(MO-SM control) is sent to the (U)SAT  3- Applet1 calls the method EnvelopeResponseHandler.postAsBERTLV() to change any incoming TP\_Destination\_Address and any RP\_Destination\_Address of the Service Center into +11 22 33 44  4- A Fetch command is sent to the (U)SAT  5- A Terminal Response command is sent to the (U)SAT  6- Delete Applet1 & Applet2  7- Install Applet3 | 1- Applet2 is suspended  2- Applet1 is triggered.  5- Applet2 execution shall continue. | 3- The U(SAT) answers 91xx to the Envelope(MO-Short Message Control)  The new TP\_Destination\_Address and RP\_Destination\_Address are retrieved. |
| 2 | Applet3 is registered on both the events EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM and EVENT\_MENU\_SELECTION.  1- Applet3 invokes the method send()and no fetch is performed  2- Envelope(MO-SM control) is sent to the (U)SAT  3- Applet3 calls the method EnvelopeResponseHandler.postAsBERTLV() to change any incoming TP\_Destination\_Address and any RP\_Destination\_Address of the Service Center into +11 22 33 44.  4- A Fetch command is sent to the (U)SAT  5- A Terminal Response command is sent to the (U)SAT | 1- Applet3 is suspended on the send() method  2- Applet3 is triggered on the EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM.  5- Applet3 execution shall continue. | 3- The U(SAT) answers 91xx to the Envelope(MO-Short Message Control)  The new TP\_Destination\_Address and RP\_Destination\_Address are retrieved. |

### 5.3.6 Toolkit installation

#### 5.3.6.1 Minimum security level

Test Area Reference: ufw\_tin\_msl

5.3.6.1.1 Conformance requirements

5.3.6.1.1.1 Normal execution

- CRRN1: The Receiving Entity shall check the Minimum Security Level during processing the security of the Command Packet.

- CRRN2: The Receiving Entity shall reject the message if the MSL check fails.

- CRRN3: If the MSL check fails, a Response Packet with the 'Insufficient Security Level' Response Status Code shall be sent if required.

- CRRN4: If the length of the Minimum Security Level field is greater than zero, the Minimum Security Level is used to specify the minimum level to be applied to Secured Packets. The first byte shall be the MSL Parameter, other bytes shall be the MSL Data.

- CRRN5: If the length of the Minimum Security Level field is zero, no minimum security level check shall be performed by the receiving entity.

5.3.6.1.2 Test area files

Test source: Test\_Ufw\_Tin\_Msl\_1.java

Test Applet: Ufw\_Tin\_Msl\_1.java

Cap file: ufw\_tin\_msl.cap

5.3.6.1.3 Test coverage

| CRR number | Test case number |
| --- | --- |
| CRRN1 | Not applicable |
| CRRN2 | 2, 4 |
| CRRN3 | 2, 4 |
| CRRN4 | 2, 4 |
| CRRN5 | 1, 4 |

5.3.6.1.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
| 1 | Installation with MSL length of 0  FORMATTED\_SMS\_PP\_ENV  1- Install (install) applet with a MSL length = 0  2- Send formatted SMS PP ENV with no RC/CC/DS, no Ciphering and counter mode 0 (not checked)  3- Send a formatted SMS PP ENV with CC, ciphering and counter mode 1 (counter available and no checking)  4- Delete the applet instance | 2- Applet is triggered  3- Applet is triggered | 1- SW = 9000 |
| 2 | Installation with correct MSL value  FORMATTED\_SMS\_PP\_ENV  1- Install (install) applet with MSL field set to 02 (CC needed).  2- Send formatted SMS PP ENV with no RC/CC/DS, no Ciphering and counter mode 0 (not checked)  3- Send a formatted SMS PP ENV with CC, no ciphering and counter mode 1 counter available and no checking)  4- Send a formatted SMS PP ENV with PoR required and no CC, ciphered with DES and counter mode 0 counter available and no checking)  5- Delete the applet instance | 2- Applet is not triggered  3- Applet is triggered  4- Applet is not triggered | 1- SW = 9000  2- SW = 61 XX, Response status code shall be '0A', insufficient security level.  or  SW = 9000  4- SW = 61 00, Response status code shall be '0A', insufficient security level  or  SW = 9000 |
| 3 | Installation with MSL length of 0  FORMATTED\_USSD  1- Install (install) applet with a MSL length = 0  2- Send formatted ÚSSD with no RC/CC/DS, no Ciphering and counter mode 0 (not checked)  3- Send a formatted USSD with CC, ciphering and counter mode 1 (counter available and no checking)  4- Delete the applet instance | 2- Applet is not triggered  3- Applet is triggered  4- Applet is not triggered | 1- SW = 9000  2- SW = 61 XX, Response status code shall be '0A', insufficient security level.  or  SW = 9000  4- SW = 61 XX, Response status code shall be '0A', insufficient security level.  or  SW = 9000 |
| 4 | Installation with correct MSL value  FORMATTED\_USSD  1- Install (install) applet with MSL field set to 02 (CC needed).  2- Send formatted SMS PP ENV with no RC/CC/DS, no Ciphering and counter mode 0 (not checked)  3- Send a formatted SMS PP ENV with CC, no ciphering and counter mode 1 counter available and no checking)  4- Send a formatted SMS PP ENV with PoR required and no CC, ciphered with DES and counter mode 0 counter available and no checking)  5- Delete the applet instance | 2- Applet is not triggered  3- Applet is triggered  4- Applet is not triggered | 1- SW = 9000  2- SW = 61 XX, Response status code shall be '0A', insufficient security level.  or  SW = 9000  4- SW = 61 00, Response status code shall be '0A', insufficient security level  or  SW = 9000 |

#### 5.3.6.2 TAR

Test Area Reference: ufw\_tin\_tar

5.3.6.2.1 Conformance requirements

5.3.6.2.1.1 Normal execution

- CRRN1: It is possible to define several TAR values at the installation of the Toolkit Application .

- CRRN2: If the length of the TAR value is zero, the TAR may be taken out of the AID if any.

- CRRN3: If the length of the TAR value is greater than zero then the application instance shall be installed with the TAR value field defined in parameter and the TAR indicated in the AID if any shall be ignored.

- CRRN4: If the TAR value(s) is already assigned on the card for the Toolkit Application instance the card shall return the status word '6A 80', incorrect parameter in data field.

- CRRN5: If the length of the TAR value(s) field is incorrect, the card shall return the status word '6A 80', incorrect parameter in data field.

5.3.6.2.2 Test area files

Test source: Test\_Ufw\_Tin\_Tar\_1.scr

Test Applet: ufw\_tin\_tar\_1.java

Cap file: ufw\_tin\_tar.cap

5.3.6.2.3 Test Coverage

| CRR number | Test case number |
| --- | --- |
| CRRN1 | 1 |
| CRRN2 | 2 |
| CRRN3 | 3 |
| CRRN4 | 4 |
| CRRN5 | 5 |

5.3.6.2.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
| 1 | Installation with several TAR value  FORMATTED\_SMS\_PP\_ENV  1- Install (install) applet with no TAR defined in the AID but TAR values set to 11 11 11, 22 22 22  2- Send formatted SMS PP ENV with TAR set to 11 11 11.  3- Send formatted SMS PP ENV with TAR set to 22 22 22.  4- Delete the applet instance. | 2- Applet is triggered  3- Applet is triggered | 1- SW = 9000 |
| 2 | Installation without TAR value in install parameter  FORMATTED*\_*SMS\_PP\_ENV  1- Install (install) applet with TAR value length set to 0 in install parameters.  2- Send formatted SMS PP ENV with the TAR value defined in applet AID.  3- Delete the applet instance | 2- Applet is triggered | 1- SW = 9000 |
| 3 | Installation with TAR value within AID different from the one define in install parameters  FORMATTED*\_*SMS\_PP\_ENV  1- Install (install) applet with applet AID TAR set to XX YY ZZ and TAR value set to 11 11 11.  2- Send formatted SMS PP ENV with the TAR value set to the one defined in applet AID.  3- Send formatted SMS PP ENV with the TAR value set to 11 11 11. | 2- Applet is not triggered.  3- Applet is triggered |  |
| 4 | Installation with TAR value already assigned  FORMATTED*\_*SMS\_PP\_ENV  1- Install (install) applet with no TAR in applet AID and TAR value in install parameters set to 11 11 11. | 1- Applet is not installed | 1- SW = 6A80 |
| 5 | Installation with incorrect TAR value length in install parameters  FORMATTED*\_*SMS\_PP\_ENV  1- Install (install) applet with no TAR in applet AID and TAR value length set to 02 and TAR value set to 11 11.  2- Install (install) applet with no TAR in applet AID and TAR value length set to 05 and TAR value set to 11 11 11,22 22. | 1- Applet is not installed    2- Applet is not installed | 1- SW = 6A80 |
| 7 | Installation with several TAR value  FORMATTED\_USSD  1- Install (install) applet with no TAR defined in the AID but TAR values set to 33 33 33, 44 44 44  2- Send formatted USSD with TAR set to 33 33 33.  3- Send formatted USSD with TAR set to 44 44 44.  4- Delete the applet instance. | 2- Applet is triggered  3- Applet is triggered | 1- SW = 9000 |
| 8 | Installation without TAR value in install parameter  FORMATTED\_USSD  1- Install (install) applet with TAR value length set to 0 in install parameters.  2- Send formatted USSD with the TAR value defined in applet AID.  3- Delete the applet instance | 2- Applet is triggered | 1- SW = 9000 |
| 9 | Installation with TAR value within AID different from the one defined in install parameters  FORMATTED\_USSD  1- Install (install) applet with applet AID TAR set to XX YY ZZ and TAR value set to 33 33 33.  2- Send formatted USSD with the TAR value set to the one defined in applet AID.  3- Send formatted USSD with the TAR value set to 33 33 33. | 2- Applet is not triggered.  3- Applet is triggered |  |
| 10 | Installation with TAR value already assigned  FORMATTED*\_*USSD  1- Install (install) applet with no TAR in applet AID and TAR value in install parameters set to 33 33 33. | 1- Applet is not installed | 1- SW = 6A80 |
| 11 | Installation with incorrect TAR value length in install parameters  FORMATTED*\_*USSD  1- Install (install) applet with no TAR in applet AID and TAR value length set to 02 and TAR value set to 11 11.  2- Install (install) applet with no TAR in applet AID and TAR value length set to 05 and TAR value set to 11 11 11,22 22. | 1- Applet is not installed    2- Applet is not installed | 1- SW = 6A80 |

#### 5.3.6.3 Access domain

Test Area Reference: ufw\_tin\_acdo

5.3.6.3.1 Conformance requirements

5.3.6.3.1.1 Normal execution

- CRRN1: The USAT framework shall grant files access to the application instance according to the USAT Framework access parameters.

- CRRN2:The access rights granted to an application and defined in the access parameter shall be independent from the access rights granted at the USAT Framework/Terminal interface level.

- CRRN3: If an application with access domain parameter set to 'FF' tries to access a file, the USAT framework shall throw an exception.

- CRRN4: An application with access domain parameter set to '00' can perform all actions on files, except the ones with NEVER access conditions.

- CRRN5: If a requested Access Domain is not supported, the card shall return the Status word '6A 80' to the Install(install) command.

- CRRN6: The card shall at least support the following Access Domain Parameter values:

- '00' : Full access to the file system

- '02' : UICC access mechanism

- 'FF' : No access to the file system

5.3.6.3.2 Test area files

Test source: Test\_Ufw\_Tin\_Acdo\_1.java

Test Applet: Ufw\_Tin\_Acdo\_1.java

Cap file: ufw\_tin\_acdo.cap

5.3.6.3.3 Test coverage

| CRR number | Test case number |
| --- | --- |
| CRRN1 | 1,2,3 |
| CRRN2 | 1 |
| CRRN3 | 2 |
| CRRN4 | 3 |
| CRRN5 | Not testable |
| CRRN6 | 1,2,3 |

5.3.6.3.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
| 1 | The access granted to an application shall be independent from the USAT Framework/Terminal interface  1- Install (install) applet with Access domain parameter set to PIN1  2- Block PIN1 by sending unsuccessful verify pin command.  3- Trigger the installed applet and try to increase EF\_CUAC file with uicc.access.increase() method.  4- Delete the applet instance. | 3- no exception is thrown |  |
| 2 | Access domain parameter set to 'FF'  1- Install (install) applet with access domain parameter set to 'FF'.  2- Trigger the installed applet and try to access EF\_TARU file with uicc.access.readBinary() method.  3- Delete the applet instance | 2- UICCException. Security\_status\_Not\_satisfied is thrown |  |
| 3 | Access domain parameter set to '00'  1- Install (install) applet with access domain parameter set to '00'.  2- Trigger the installed applet and try to access EF\_TARU file using uicc.access.readBinary() method.  3- Trigger the applet and try to access the EF\_TNU file using uicc.access.updateBinary() method.  4- Delete the applet instance | 2- No exception is thrown.  3- UICCException. Security\_status\_Not\_satisfied is thrown |  |

### 5.3.7 Other parts transferred to (U)SAT framework from API

#### 5.3.7.1 A handler is a temporary JCRE Entry Point object

Test Area Reference: Ufw\_Api\_Hepo

5.3.7.1.1 Conformance requirement

5.3.7.1.1.1 Normal execution

- CRRN1: The *USATEnvelopeHandler* is a Temporary JCRE Entry Point Object (see Java Card 2.2.1 Runtime Environment (JCRE) Specification [13]).

5.3.7.1.1.2 Parameter errors

No requirements.

5.3.7.1.1.3 Context errors

No requirements.

5.3.7.1.2 Test area files

Test Source: Test\_Ufw\_Api\_Hepo.java

Test Applet: Ufw\_Api\_Hepo\_1.java

Cap File: Ufw\_Api\_Hepo.cap

5.3.7.1.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| CRRN1 | 1, 2 |

5.3.7.1.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
| 1 | Storage in a static object  Call USATEnvelopeHandler.getTheHandler() method and store it in a static field of the toolkit applet | SecurityException is thrown |  |
| 2 | Storage in a non-static object  Call USATEnvelopeHandler.getTheHandler() method and store it in a non-static field of the toolkit applet | SecurityException is thrown |  |

### 5.3.8 Framework security management

Security parameters

The table that follows contains the security parameters that shall be used when the TS 31.115[10] security is required in the test cases developed in the current clause.

|  |  |
| --- | --- |
| Parameter | Value in hexadecimal |
| KIC | Value as described in the TS 31.115 [10] (recommended value: 15) |
| KID | Value as described in the TS 31.115 [10] (recommended value: 15) |
| CNTR | 00 00 00 00 01 |
| Key for ciphering | Corresponding to KIC (recommended value: 01 41 42 7F DA E8 91 A7 02 41 42 7F DA E8 91 A7) |
| Key for RC/CC/DS | Corresponding to KID (recommended value: 01 23 45 67 89 AB CD EF EF CD AB 89 67 45 23 01) |

If a parameter is not listed explicitly in the above table, the default values of Annex B apply.

#### 5.3.8.1 Input data

Test Area Reference: ufw\_fws\_inda

5.3.8.1.1 Conformance requirements

5.3.8.1.1.1 Normal execution

- CRRN1: If the USAT Framework receives an envelope APDU containing a Short Message Point to Point formatted according to TS 31.115 [10], it shall verify the security of the SMS TPDU and trigger the applet registered with the corresponding TAR.

- CRRN2: The toolkit applet will only be triggered if the TAR is known and the security verified.

- CRRN3: If the USAT Framework receives an envelope APDU containing a Short message Cell Broadcast formatted according to TS 31.115 [10], it shall verify the security of the cell broadcast page and trigger the applet registered with the corresponding TAR.

- CRRN4: If the USAT Framework receives an Update Record EFSMS instruction formatted according to TS 31.115 [10], it shall verify the security of the SMS and trigger the applet registered with the corresponding TAR.

- CRRN5: The USAT Framework shall provide the input data deciphered.

5.3.8.1.1.2 Parameter errors

No requirements.

5.3.8.1.1.3 Context errors

No requirements.

5.3.8.1.2 Test area files

Test source: Test\_ufw\_fws\_inda\_1.java

Test Applet: ufw\_fws\_inda\_1.java

ufw\_fws\_inda\_2.java

ufw\_fws\_inda\_3.java

ufw\_fws\_inda\_4.java

ufw\_fws\_inda\_5.java

ufw\_fws\_inda\_6.java

Cap file: ufw\_fws\_inda.cap

5.3.8.1.3 Test coverage

| CRR Number | Test Case Number |
| --- | --- |
| CRRN1 | 1, 2, 3 |
| CRRN2 | 3,6,9 |
| CRRN3 | 4, 5, 6 |
| CRRN4 | 7,8,9 |
| CRRN5 | 1,2,4,5,7,8 |

5.3.8.1.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
| 1 | Framework checks the Cryptographic checksum and deciphers the data  SMS-PP  Applet1 is loaded and installed  1-Envelope(SMS-PP) single and formatted is sent to the USAT Framework with this features:  Ciphering;  Cryptographic checksum;  No proof of receipt;  TAR of Applet1;  Data = 01  2- Short Message concatenated and formatted is sent to the USAT Framework by an Envelope (SMS PP)with these features:  Ciphering;  Cryptographic checksum;  No proof of receipt;  TAR of Applet1;  Data length is 150. | 1- Applet1 is triggered and the value integrity is checked.  2- Applet1 is triggered and the value integrity is checked | 1- The USAT Framework answers to the Envelope with status words 9000  2- The USAT Framework answers to the Envelope with status words 9000 |
| 2 | Triggering two different applets with different security  Applet2 is installed  1-Envelope(SMS-PP) single and formatted is sent to the USAT Framework with this features:  Ciphering;  Cryptographic checksum;  No proof of receipt;  TAR of Applet1  Data = 03  2- Short Message concatenated and formatted is sent to the USAT Framework by an Envelope (SMS PP)with these features:  Ciphering;  Cryptographic checksum;  No proof of receipt;  TAR of Applet1  Data length = 150  3-Envelope(SMS-PP) single and formatted is sent to the USAT Framework with this features:  No ciphering;  No cryptographic checksum;  No proof of receipt;  TAR of Applet2  Data = 05  4- Short Message concatenated and formatted is sent to the USAT Framework by an Envelope (SMS PP)with these features:  No ciphering;  No cryptographic checksum;  No proof of receipt;  TAR of Applet2  Data length = 150. | 1- Applet1 is triggered and the value integrity is checked  2- Applet1 is triggered and the value integrity is checked  3- Applet2 is triggered and the value integrity is checked  4- Applet2 is triggered and the value integrity is checked | 1- The USAT Framework answers to the Envelope with status words 9000  2- The USAT Framework answers to the Envelope with status words 9000  3- The USAT Framework answers to the Envelope with status words 9000  4- The USAT Framework answers to the Envelope with status words 9000 |
| 3 | Envelope(SMS-PP) formatted with wrong cryptographic checksum  1-Envelope 03.48 single and formatted is sent to the USAT Framework with this features:  No ciphering;  Wrong cryptographic checksum;  No proof of receipt;  TAR of Applet1  Data = 07  2- Short Message concatenated and formatted is sent to the USAT Framework by an Envelope (SMS PP)with these features:  No ciphering;  Wrong cryptographic checksum;  No proof of receipt;  TAR of Applet1  Data length = 150 | 1- No applet is triggered.  2- No applet is triggered. | 1- The USAT Framework answers to the Envelope with status words 9000 |
| 4 | **Framework checks the Cryptographic checksum and deciphers the data**  Applet3 is loaded and installed  1-Envelope(SMS-CB) formatted is sent to the USAT Framework with this features:  Ciphering;  Cryptographic checksum;  No proof of receipt;  Data = 01 | 1- Applet3 is triggered and the value integrity is checked | 1- The USAT Framework answers to the Envelope with status words 9000 |
| 5 | **Triggering two different applets with different security on Envelope(SMS-CB) formatted**  Applet4 is installed  1-Envelope(SMS-CB) formatted is sent to the USAT Framework with this features:  Ciphering;  Cryptographic checksum;  No proof of receipt;  TAR of Applet3  Data = 02  2-Envelope(SMS-CB) formatted is sent to the USAT Framework with this features:  No ciphering;  No cryptographic checksum;  No proof of receipt;  TAR of Applet4  Data = 03 | 1- Applet3 is triggered and the value integrity is checked  2- Applet4 is triggered and the value integrity is checked | 1- The USAT Framework answers to the Envelope with status words 9000  2- The USAT Framework answers to the Envelope with status words 9000 |
| 6 | Envelope(SMS-CB) formatted with wrong cryptographic checksum  No ciphering;  Wrong Cryptographic checksum;  No proof of receipt;  TAR of Applet3  Data = 04 | No applet is triggered | 1- The USAT Framework answers to the Envelope with status words 9000 |
| 7 | Framework checks the Cryptographic checksum and deciphers the data  Applet5 is installed  1- Short Message single and formatted is sent to the USAT Framework by Update Record EFSMS instruction with these features:  Ciphering;  Cryptographic checksum;  No proof of receipt;  TAR of Applet5;  Data = 01  2- Short Message concatenated and formatted is sent to the USAT Framework by Update Record EFSMS instruction with these features:  Ciphering;  Cryptographic checksum;  No proof of receipt;  TAR of Applet5;  Data length = 150. | 1- Applet5 is triggered and the value integrity is checked.  2- Applet5 is triggered and the value integrity is checked | 1- The USAT Framework answers to the Update Record EFSMS instruction with status words 9000  2- The USAT Framework answers to the Update Record EFSMS instruction with status words 9000 |
| 8 | Triggering two different applets with different security  Applet6 is installed  1- Short Message single and formatted is sent to the USAT Framework by Update Record EFSMS instruction with these features:  Ciphering;  Cryptographic checksum;  No proof of receipt;  TAR of Applet5  Data = 03  2- Short Message concatenated and formatted is sent to the USAT Framework by Update Record EFSMS instruction with these features:  Ciphering;  Cryptographic checksum;  No proof of receipt;  TAR of Applet5  Data length = 150.  3- Short Message single and formatted is sent to the USAT Framework by Update Record EFSMS instruction with these features:  No ciphering;  No cryptographic checksum;  No proof of receipt;  TAR of Applet6;  Data = 05  4- Short Message concatenated and formatted is sent to the USAT Framework by Update Record EFSMS instruction with these features:  No ciphering;  No cryptographic checksum;  No proof of receipt;  TAR of Applet6;  Data length = 150. | 1- Applet5 is triggered and the value integrity is checked.  2- Applet5 is triggered and the value integrity is checked.  3- Applet6 is triggered and the value integrity is checked.  4- Applet6 is triggered and the value integrity is checked. | 1- The USAT Framework answers to the Update Record EFSMS instruction with status words 9000  2- The USAT Framework answers to the Update Record EFSMS instruction with status words 9000  3- The USAT Framework answers to the Update Record EFSMS instruction with status words 9000  4- The USAT Framework answers to the Update Record EFSMS instruction with status words 9000 |
| 9 | Update Record EFSMS instruction formatted with wrong cryptographic checksum  1- Short Message single and formatted is sent to the USAT Framework by Update Record EFSMS instruction with these features: No ciphering;  Wrong Cryptographic checksum;  No proof of receipt;  TAR of Applet5  Data = 07  2- Short Message concatenated and formatted is sent to the USAT Framework by Update Record EFSMS instruction with these features:  No ciphering;  Wrong Cryptographic checksum;  No proof of receipt;  TAR of Applet5  Data length = 150 | 1- No applet is triggered.  2- No applet is triggered. | 1- The USAT Framework answers to the Update Record EFSMS instruction with status words 9000  2- The USAT Framework answers to the Update Record EFSMS instruction with status words 9000 |
| 10 | Framework checks the Cryptographic checksum and deciphers the data  USSD  Applet1 is loaded and installed  1-Envelope(USSD) single and formatted is sent to the USAT Framework with this features:  Ciphering;  Cryptographic checksum;  No proof of receipt;  TAR of Applet1;  Data = 01  2- USSD Message concatenated and formatted is sent to the USAT Framework by an Envelope (USDD)with these features:  Ciphering;  Cryptographic checksum;  No proof of receipt;  TAR of Applet1;  Data length is 150. | 1- Applet1 is triggered and the value integrity is checked.  2- Applet1 is triggered and the value integrity is checked | 1- The USAT Framework answers to the Envelope with status words 9000  2- The USAT Framework answers to the Envelope with status words 9000 |
| 11 | **Triggering two different applets with different security**  **USSD**  Applet2 is installed  1-Envelope(USSD) single and formatted is sent to the USAT Framework with this features:  Ciphering;  Cryptographic checksum;  No proof of receipt;  TAR of Applet1  Data = 03  2- Short Message concatenated and formatted is sent to the USAT Framework by an Envelope (USSD)with these features:  Ciphering;  Cryptographic checksum;  No proof of receipt;  TAR of Applet1  Data length = 150  3-Envelope(USSD) single and formatted is sent to the USAT Framework with this features:  No ciphering;  No cryptographic checksum;  No proof of receipt;  TAR of Applet2  Data = 05  4- Concatenated and formatted USSD is sent to the USAT Framework by an Envelope (USSD)with these features:  No ciphering;  No cryptographic checksum;  No proof of receipt;  TAR of Applet2  Data length = 150. | 1- Applet1 is triggered and the value integrity is checked  2- Applet1 is triggered and the value integrity is checked  3- Applet2 is triggered and the value integrity is checked  4- Applet2 is triggered and the value integrity is checked | 1- The USAT Framework answers to the Envelope with status words 9000  2- The USAT Framework answers to the Envelope with status words 9000  3- The USAT Framework answers to the Envelope with status words 9000  4- The USAT Framework answers to the Envelope with status words 9000 |
| 12 | USSD formatted with wrong cryptographic checksum  USSD  1-Formatted USSD is sent to the USAT Framework with this features:  No ciphering;  Wrong cryptographic checksum;  No proof of receipt;  TAR of Applet1  Data = 07  2- USSD concatenated and formatted is sent to the USAT Framework by an Envelope (USSD)with these features:  No ciphering;  Wrong cryptographic checksum;  No proof of receipt;  TAR of Applet1  Data length = 150 | 1- No applet is triggered.  2- No applet is triggered. | 1- The USAT Framework answers to the Envelope with status words 9000 |
| 13 | **Framework checks the Cryptographic checksum and deciphers the data**  **USSD**  Applet3 is loaded and installed  1-USSD formatted is sent to the USAT Framework with this features:  Ciphering;  Cryptographic checksum;  No proof of receipt;  Data = 01 | 1- Applet3 is triggered and the value integrity is checked | 1- The USAT Framework answers to the Envelope with status words 9000 |

#### 5.3.8.2 Output data

Test Area Reference: ufw\_fws\_ouda

5.3.8.2.1 Conformance requirements

5.3.8.2.1.1 Normal execution

- CRRN1: The USAT Framework Toolkit Framework shall secure and send the response packet.

5.3.8.2.1.2 Parameters errors

No requirements.

5.3.8.2.1.3 Context errors

No requirements.

5.3.8.2.2 Test Area Files

Test source: Test\_Ufw\_Fws\_Ouda\_1.java

Test Applet: Ufw\_Fws\_Ouda\_1.java

Cap file: Ufw\_Fws\_Ouda.cap

5.3.8.2.3 Test coverage

| CRR Number | Test Case Number |
| --- | --- |
| CRRN1 | 1, 2, 3, 4 |

5.3.8.2.4 Test procedure

| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
| --- | --- | --- | --- |
| 1 | Envelope(SMS-PP) formatted  Ciphering;  Cryptographic checksum;  proof of receipt response shall be sent using SMS-Deliver-Report;  no security applied to proof of receipt  Data in plain text = "APPLET1" | The applet is triggered and sends a "Display Text" proactive command with the data received in the Envelope. | The USAT Framework answers to the Envelope with a PoR which is retrieved and checked.  The PoR has no application data.  The USAT Framework answers with status words 91xx to issue a Display Text "APPLET1". |
| 2 | Envelope(SMS-PP) formatted  Ciphering;  Cryptographic checksum;  proof of receipt response shall be sent using SMS-Deliver-Report;  no security applied to proof of receipt  Data in plain text = "APPLET1" | The applet posts application data. It does not call the ProactiveHandler.send() method | The USAT Framework answers to the Envelope with a PoR which is retrieved and checked.  The PoR has the application data posted by the application. |
| 3 | Envelope(SMS-PP) formatted  Ciphering;  Cryptographic checksum;  proof of receipt response shall be sent using SMS-Deliver-Report;  no security applied to proof of receipt  Data in plain text = "TEST" | The applet posts application data and calls the ProactiveHandler.send() method to send a "Display Text" proactive command with the data received in the Envelope. | The USAT Framework answers to the Envelope with a PoR which is retrieved and checked. The PoR has the application data posted by the application.  The USAT Framework answers with status words 91xx to issue the Display Text "TEST". |
| 4 | Envelope(SMS-PP) formatted  Ciphering;  Cryptographic checksum;  proof of receipt response shall be sent using SMS-Deliver-Report;  proof of receipt shall be ciphered  Data in plain text = "TEST" | The applet posts application data and calls the ProactiveHandler.send() method to send a "Display Text" proactive command with the data received in the Envelope. | The USAT Framework answers to the Envelope with a PoR which is retrieved and checked. The PoR has the application data posted by the application.  The USAT Framework answers with status words 91xx to issue the Display Text "TEST". |
| 5 | Envelope(USSD) formatted  Ciphering;  Cryptographic checksum;  proof of receipt response shall be sent and no security shall be applied  Data in plain text = "APPLET1" | The applet is triggered and sends a "Display Text" proactive command with the data received in the Envelope. | The USAT Framework answers to the Envelope with a PoR which is retrieved and checked.  The PoR has no application data.  The USAT Framework answers with status words 91xx to issue a Display Text "APPLET1". |
| 6 | Envelope(USSD) formatted  Ciphering;  Cryptographic checksum;  proof of receipt response shall be sent using SMS-Deliver-Report;  no security applied to proof of receipt  Data in plain text = "APPLET1" | The applet posts application data. It does not call the ProactiveHandler.send() method | The USAT Framework answers to the Envelope with a PoR which is retrieved and checked.  The PoR has the application data posted by the application. |
| 7 | Envelope(USSD) formatted  Ciphering;  Cryptographic checksum;  proof of receipt response shall be sent and no security applied to proof of receipt  Data in plain text = "TEST" | The applet posts application data and calls the ProactiveHandler.send() method to send a "Display Text" proactive command with the data received in the Envelope. | The USAT Framework answers to the Envelope with a PoR which is retrieved and checked. The PoR has the application data posted by the application.  The USAT Framework answers with status words 91xx to issue the Display Text "TEST". |
| 8 | Envelope(USSD) formatted  Ciphering;  Cryptographic checksum;  proof of receipt response shall be sent and it shall be ciphered  Data in plain text = "TEST" | The applet posts application data and calls the ProactiveHandler.send() method to send a "Display Text" proactive command with the data received in the Envelope. | The USAT Framework answers to the Envelope with a PoR which is retrieved and checked. The PoR has the application data posted by the application.  The USAT Framework answers with status words 91xx to issue the Display Text "TEST". |

### 5.3.9 Concatenated SMS

#### 5.3.9.1 Concatenation processing

Test Area Reference: Ufw\_Csm\_Proc

5.3.9.1.1 Conformance requirements:

5.3.9.1.1.1 Normal execution

- CRRN1: When a Short Message is received as a Concatenated Short Message as defined in TS 23.040 [11], it is the responsibility of the (U)SAT Framework to link single Short Messages to reassemble the original message before any further processing.

- CRRN2: The concatenation control headers, used to reassemble the short messages in the correct order, shall not be present in the SMS TPDU.

- CRRN3: The TP-elements of the SMS TPDU and the Address (TS-Service-Centre-Address) shall correspond to the ones in the last received Short Message (independently of the Sequence number of Information-Element-Data).

- CRRN4: The original Short Message shall be placed in one SMS TPDU TLV (with TP-UDL field coded on one octet) included in the USATEnvelopeHandler.

- CRRN5: The (U)SAT Framework shall be able to process messages with the following properties as a minimum requirement:

- the Information Element Identifier is equal to the 8-bit reference number

- it contains uncompressed 8 bit data or uncompressed UCS2 data

#### 5.3.9.2 Test area files

Test Source: Test\_Ufw\_Csm\_Proc.java

Test Applet: Ufw\_Csm\_Proc\_1.java

Cap File: ufw\_csm\_proc.cap

#### 5.3.9.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | 1,2, 3, 4, , 7, 8, 9,10, , 13, 14, 15, 18, 19, 21, 22 |
| N2 | 5,17 |
| N3 | 8,11, 20, 19 |
| N4 | 5,17 |
| N5 | 12,24 |

#### 5.3.9.4 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
|  | Applet registration to EVENT\_FORMATTED\_SMS\_PP\_ENV and triggering  Applet is registered to EVENT\_FORMATTED\_SMS\_PP\_ENV and EVENT\_UNFORMATTED\_SMS\_PP\_ENV events  A concatenated formatted SMS\_PP is sent to the (U)SIM (composed of three segments). |  |  |
| 1 | The second segment of a concatenated short message is sent to the (U)SIM. | Applet is not triggered. |  |
| 2 | The first segment of the concatenated short message is sent to the (U)SIM. | Applet is not triggered. |  |
| 3 | The third segment of the concatenated short message is send to the (U)SIM | Applet is triggered. |  |
| 4 | Call USATEnvelopeHandlerSystem.getTheHandler(). | No exception is thrown. |  |
| 5 | Call the USATEnvelopeHandler.findTLV() to select the Dev Id, the address and the TPDU TLV and the USATEnvelopeHandler.compareValue() to check each content. | Check that the message has been re-assembled in the correct order. Check that TP-UDL field is coded in one octet. Check that the concatenation control header is not present in the message. Check the integrity of the message. |  |
| 6 | A new concatenated formatted short message is sent to the USIM composed of three segments. The Address fields of the first, second and the third segment are different. | Applet is triggered. |  |
| 7 | Call USATEnvelopeHandlerSystem.getTheHandler(). | No exception is thrown. |  |
| 8 | Call USATEnvelopeHandlerSystem.findTLV() to select the address TLV and the USATEnvelopeHandler.compareValue() to check its content. | Check that the address field of the message is equal to the address field of the third segment. |  |
| 9 | A new concatenated formatted short message is sent to the (U)SIM composed of three segments. Some TP elements of the TPDU of the first, second and third segment are different among themselves. | Applet is triggered. |  |
| 10 | Call USATEnvelopeHandlerSystem.getTheHandler(). | No exception is thrown. |  |
| 11 | Call USATEnvelopeHandler.findTLV() to select the TP\_DU TLV and USATEnvelopeHandler.compareValue() to check its TP elements. | Check that the TP elements of the message are equal to the ones of the third segment. |  |
| 12 | Send a concatenated formatted short message (composed of 3 segments) with uncompressed 8 bit data. | Applet is triggered. |  |
|  | Applet registration to EVENT\_UNFORMATTED\_SMS\_PP\_ENV and triggering  Same test as above but with an unformatted SMS\_PP envelope.  A concatenated unformatted SMS\_PP is sent to the (U)SIM (composed of three segments). |  |  |
| 13 | The second segment of a concatenated short message is sent to the (U)SIM. | Applet is not triggered. |  |
| 14 | The first segment of the concatenated short message is sent to the (U)SIM. | Applet is not triggered. |  |
| 15 | The third segment of the concatenated short message is sent to the (U)SIM. | Applet is triggered. |  |
| 16 | Call USATEnvelopeHanlderSystem.getTheHandler() | No exception is thrown. |  |
| 17 | Call USATEnvelopeHandler.findTLV() to select the Dev Id, the address and the TPDU TLV and the USATEnvelopeHandler.compareValue() to check each content. | Check that the message has been reassembled in the correct order. Check that TP-UDL field is coded one octet. Check that the concatenation control header is not present in the message. Check the integrity of the message. |  |
| 18 | A new concatenated formatted short message is sent to the (U)SIM composed of three segments. The Address field of the first segment is different from the address field in the second segment. | Applet is triggered. |  |
| 19 | Call USATEnvelopeHandlerSystem.getTheHandler(). | No exception is thrown. |  |
| 20 | Call USATEnvelopeHandler.findTLV() to select the address TLV and the USATEnvelopeHandler.compareValue() to check its content. | Check that the address field of the message is equal to the address field of the second segment. |  |
| 21 | A new concatenated unformatted short message is sent to the (U)SIM composed of two segments. Some TP\_elements of the TPDU of the first, second and third segment are different. | Applet is triggered. |  |
| 22 | Call USATEnvelopeHandlerSystem.getTheHandler() | No exception is thrown. |  |
| 23 | Call USATEnvelopeHandler.findTLV()to select the TPDU TLV and the USATEnvelopeHandler.compareValue() to check its TP elements. | Check that the TP elements of the message are equal to the ones of the third segment. |  |
| 24 | Send a concatenated unformatted short message (composed of 3 segments) with uncompressed UCS2 data. | Applet is triggered. |  |

### 5.3.10 Cell Broadcast Service

#### 5.3.10.1 Multiple message reassembling

Test Area Reference: Ufw\_Cbs\_Mmra

5.3.10.1.1 Conformance requirements:

5.3.10.1.1.1 Normal execution

- CRRN1: When a Cell Broadcast Message is received as multiple pages as defined in TS 23.041 [6], it is the responsibility of the (U)SAT Framework to link single pages together to re- assemble the original message before any further processing.

- CRRN2: The original Cell Broadcast message shall be placed in one Cell Broadcast page TLV included in the USATEnvelopeHandler.

- CRRN3: The message parameters shall correspond to the ones in the last received Cell Broadcast page (independently of the Page Parameter).

#### 5.3.10.2 Test area files

Test Source: Test\_Ufw\_Cbs\_Mmra.java

Test Applet: Ufw\_Cbs\_Mmra\_1.java

Cap File: Ufw\_Cbs\_Mmra.cap

#### 5.3.10.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | 1 – 10 |
| N2 | 5, 10 |
| N3 | 5, 10 |

5.3.10.4 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
|  | Applet registration to EVENT\_FORMATTED\_SMS\_CB and triggering  Applet is registered to EVENT\_FORMATTED\_SMS\_CB\_ENV and EVENT\_UNFORMATTED\_SMS\_CB\_ENV events  A multi page formatted SMS\_CB is sent to the (U)SIM (composed of three segments). |  |  |
| 1 | The first segment of a multi page formatted CB short message is sent to the (U)SIM. | Applet is not triggered. |  |
| 2 | The second segment of a multi page formatted CB short message is sent to the USIM. | Applet is not triggered. |  |
| 3 | The third segment of the multi page formatted CB short message is sent to the USIM. | Applet is triggered. |  |
| 4 | Call USATEnvelopeHandlerSystem.getTheHandler(). | No exception is thrown. |  |
| 5 | Call USATEnvelopeHandler.compareValue() to check message parameter, message integrity and message content. | The message is re- assembled in the correct order with Message Parameter of the third SMS\_CB which was sent at last. |  |
|  | Applet registration to EVENT\_UNFORMATTED\_SMS\_CB and triggering  A multi page unformatted SMS\_CB is sent to the (U)SIM (composed of three segments). |  |  |
| 6 | The second segment of a multi page unformatted SMS\_CB is sent to the (U)SIM. | Applet is not triggered. |  |
| 7 | The third segment of a multi page unformatted SMS\_CB is sent to the (U)SIM. | Applet is not triggered. |  |
| 8 | The first segment of a concatenated unformatted SMS\_CB is sent to the (U)SIM. | Applet is triggered. |  |
| 9 | Call USATEnvelopeHandlerSystem.getTheHandler(). | No exception is thrown. |  |
| 10 | Call USATEnvelopeHandler.compareValue() to check message parameter, message integrity and message content. | The message is re- assembled in the correct order with Page Parameter of first SMS\_CB which was sent at last. |  |

### 5.3.11 Concatenated USSD

#### 5.3.11.1 Concatenation processing

Test Area Reference: Ufw\_Fusd\_Proc

5.3.11.1.1 Conformance requirements:

5.3.11.1.1.1 Normal execution

- CRRN1: When a USSD Message is received as concatenated as defined in TS 31.115 [X], it is the responsibility of the (U)SAT Framework to link single USSD Messages together to re‑assemble the original message before any further processing.

- CRRN2: The original USSD message shall be placed in one USSD String TLV included in the USATEnvelopeHandle.

- CRRN3: The USSD String parameters (DCS, PFI, CCF) shall correspond to the ones in the last received USSD String (independently of the CCF Sequence number).

- CRRN4: The (U)SAT Framework shall be able to process messages with the following properties as a minimum requirement:

- the Information Element Identifier is equal to the 8-bit reference number

- it contains uncompressed 8 bit data or uncompressed UCS2 data

#### 5.3.11.2 Test area files

Test Source: Test\_Ufw\_Fusd\_Proc.java

Test Applet: Ufw Fusd Proc\_1.java

Cap File: ufw fusd proc.cap

#### 5.3.11.3 Test coverage

|  |  |
| --- | --- |
| CRR number | Test case number |
| N1 | 1 to 9, |
| N2 | 1 to 9 |
| N3 | 6, 10 to 18 |
| N4 | 9,18 |

#### 5.3.11.4 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Description | API/(U)SAT Framework Expectation | APDU Expectation |
|  | Applet registration to EVENT\_FORMATTED\_USSD and triggering  Applet is registered to EVENT\_FORMATTED\_USSD and EVENT\_UNFORMATTED\_USSD events  A concatenated formatted USSD is sent to the (U)SIM (composed of three segments). |  |  |
| 1 | The second segment of a concatenated USSD is sent to the (U)SIM. | Applet is not triggered. |  |
| 2 | The first segment of the concatenated USSD is sent to the (U)SIM. | Applet is not triggered. |  |
| 3 | The third segment of the concatenated USSD is send to the (U)SIM | Applet is triggered. |  |
| 4 | Call USATEnvelopeHandlerSystem.getTheHandler(). | No exception is thrown. |  |
| 5 | Call the USATEnvelopeHandler.findTLV() to select the Dev Id, the alpha id and the USSD String and the USATEnvelopeHandler.compareValue() to check each content. | Check that the message has been re-assembled in the correct order. Check that TP-UDL field is coded in one octet. Check that the concatenation control header is not present in the message. Check the integrity of the message. |  |
| 6 | A new concatenated formatted USSD is sent to the (U)SIM composed of three segments. DCS, PFI, CCF of the of the first, second and third segment are different among themselves. | Applet is triggered. |  |
| 7 | Call USATEnvelopeHandlerSystem.getTheHandler(). | No exception is thrown. |  |
| 8 | Call USATEnvelopeHandler.findTLV() to select the TP\_DU TLV and USATEnvelopeHandler.compareValue() to check its TP elements. | Check that the TP elements of the message are equal to the ones of the third segment. |  |
| 9 | Send a concatenated USSD (composed of 3 segments) with uncompressed 8 bit data. | Applet is triggered. |  |
|  | Applet registration to EVENT\_UNFORMATTED\_USSD and triggering  Same test as above but with an unformatted USSD  A concatenated unformatted SMS\_PP is sent to the (U)SIM (composed of three segments). |  |  |
| 10 | The second segment of a concatenated USSD message is sent to the (U)SIM. | Applet is not triggered. |  |
| 11 | The first segment of the concatenated USSD message is sent to the (U)SIM. | Applet is not triggered. |  |
| 12 | The third segment of the concatenated USSD message is sent to the (U)SIM. | Applet is triggered. |  |
| 13 | Call USATEnvelopeHanlderSystem.getTheHandler() | No exception is thrown. |  |
| 14 | Call USATEnvelopeHandler.findTLV() to select the Dev Id, USSD String TLV and the USATEnvelopeHandler.compareValue() to check each content. | Check that the message has been reassembled in the correct order. Check that TP-UDL field is coded one octet. Check that the concatenation control header is not present in the message. Check the integrity of the message. |  |
| 15 | A new concatenated unformatted USSD is sent to the (U)SIM composed of two segments. Some TP\_elements of the TPDU of the first, second and third segment are different. | Applet is triggered. |  |
| 16 | Call USATEnvelopeHandlerSystem.getTheHandler() | No exception is thrown. |  |
| 17 | Call USATEnvelopeHandler.findTLV()to select the TPDU TLV and the USATEnvelopeHandler.compareValue() to check its TP elements. | Check that the TP elements of the message are equal to the ones of the third segment. |  |
| 18 | Send a concatenated unformatted USSD (composed of 3 segments) with uncompressed UCS2 data. | Applet is triggered. |  |

Annex A (normative):  
Class, methods and USATFramework tests acronyms

# A.1 Toolkit part

|  |  |
| --- | --- |
| USATEnvelopeHandler | Ueh |
| USATEnvelopeHandlerSystem | Ues |
| ToolkitRegistry | Tkr |

## A.1.1 USATEnvelopeHandler interface

|  |  |
| --- | --- |
| Method name | Acronyms |
| short getSecuredDataLength() | Gsdl |
| short getSecuredDataOffset() | Gsdo |
| short getShortMessageLength() | Gsml |
| short getShortMessageOffset() | Gsmo |
| short getTPUDLOffset() | Gtpo |
| short getUserDataLength() | Gudl |
|  |  |
| Inherited method name: EnvelopeHandler | Acronyms |
| byte getChannelIdentifier() | Gcid |
| short getChannelStatus(byte channelIdentifier) | Gcst |
| byte getItemIdentifier() | Giid |
|  |  |
| Inherited method name: BERTLVViewHandler |  |
| short getSize() | Gtsz |
| byte getTag() | Gttg |
|  |  |
| Inherited method name: ViewHandler |  |
| byte compareValue(short valueOffset, byte[] compareBuffer, short compareOffset, short compareLength) | Cprv |
| short copy(byte[] dstBuffer, short dstOffset, short dstLength) | Copy |
| short copyValue(short valueOffset, byte[] dstBuffer, short dstOffset, short dstLength) | Cpyv |
| byte findAndCompareValue(byte tag, byte[] compareBuffer, short compareOffset) | Facrb\_Bs |
| byte findAndCompareValue(byte tag, byte occurrence, short valueOffset, byte[] compareBuffer, short compareOffset, short compareLength) | Facrbbs\_Bss |
| short findAndCopyValue(byte tag, byte[] dstBuffer, short dstOffset) | Facyb\_Bs |
| short findAndCopyValue(byte tag, byte occurrence, short valueOffset, byte[] dstBuffer, short dstOffset, short dstLength) | Facybbs\_Bss |
| byte findTLV(byte tag, byte occurrence) | Find |
| short getCapacity() | Gcap |
| short getLength() | Glen |
| byte getValueByte(short valueOffset) | Gvby |
| short getValueLength() | Gvle |
| short getValueShort(short valueOffset) | Gvsh |

## A.1.2 USATEnvelopeHandlerSystem method

|  |  |
| --- | --- |
| Method Name | Acronyms |
| public static USATEnvelopeHandler getTheHandler() | Gthd |

## A.1. 3 ToolkitRegistry methods

|  |  |
| --- | --- |
| Method Name | Acronyms |
| void clearEvent(short event) | Cevt |
| boolean isEventSet(short event) | Ievs |
| void setEvent(short event) | Sevt |
| void setEventList(short[] eventList, short offset, short length) | Sevl |

# A.2 Acronyms for USATFramework tests

|  |  |
| --- | --- |
| Minimum handler availability | Mha |
| Handler integrity | Hin |
| Applet triggering | Apt |
| Exception handling | Exh |
| Envelope response posting | Erp |
| Toolkit installation | Tin |
| Other parts transferred from API to CAT RE | Api |
| Framework Security | Ufs |
| Concatenated SMS | Csm |
| Cell Broadcast Service | Cbs |

## A.2.1 Minimum handler availability

|  |  |
| --- | --- |
| Test Area within the chapter | Acronyms |
| ProactiveHandler | Pahd |
| ProactiveResponseHandler | Prhd |
| EnvelopeHandler | Enhd |
| EnvelopeResponseHandler | Erhd |
| USATEnvelopeHandler | Uehd |
| Applet triggering with ongoing proactive session | Rent |

## A.2.2 Handler integrity

|  |  |
| --- | --- |
| Test Area within the chapter | Acronyms |
| ProactiveResponseHandler | Prhd |
| EnvelopeHandler | Enhd |
| USATEnvelopeHandler | Uehd |

## A.2.3 Applet triggering

|  |  |
| --- | --- |
| Test Area within the chapter | Acronyms |
|  |  |

## A.2.4 Exception handling

|  |  |
| --- | --- |
| Method Name | Acronyms |
| General Behaviour | Genb |
| Interaction with Multiple Triggering | Imtg |

## A.2.5 Envelope response posting

|  |  |
| --- | --- |
| Method Name | Acronyms |
| EVENT\_MO\_SHORT\_MESSAGE\_CONTROL\_BY\_SIM | Emcn |

## A.2.6 Toolkit installation

|  |  |
| --- | --- |
| Method Name | Acronyms |
| Access Domain | Acdo |
| Minimum Security Level | Mslv |
| TAR Value(s) of the Toolkit Application instance | Tarv |

## A.2.7 Other parts transferred from API to CAT RE

|  |  |
| --- | --- |
| Method Name | Acronyms |
| A handler is a temporary JCRE Entry Point object | Hepo |

## A.2.8 Framework security

|  |  |
| --- | --- |
| Method Name | Acronyms |
| Input Data | Inda |
| Output Data | Ouda |

## A.2.9 Concatenated SMS

|  |  |
| --- | --- |
| Method Name | Acronyms |
| Concatenation processing | Proc |

## A.2.10 Cell Broadcast Service

|  |  |
| --- | --- |
| Method Name | Acronyms |
| Multiple message reassembling | Mmra |

Annex B (normative):  
Global prepersonalization

The file system used to pass the test suite is described in ETSI TS 102 268 [15] Annex B, with one exception for file EFSMS.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Identifier** | **Description** | **Special Notes** |
| EFSMS | 6F3C | 1st record: 00 FF … FF(length 176)  2nd record: 00 FF … FF(length 176)  3rd record: 00 FF … FF(length 176) | The file EFSMS should be present under both MF/DFTELECOM and directly under ADF1. One of these two EFSMS files must be linked to the other one. |

Additionally, the following values must be used:

|  |  |
| --- | --- |
| Parameter | Value in hexadecimal |
| KIC | Value as described in the TS 31.115 [10] (recommended value: 15) |
| KID | Value as described in the TS 31.115 [10] (recommended value: 15) |
| CNTR | 00 00 00 00 01 |
| Key for ciphering | Corresponding to KIC (recommended value: 01 41 42 7F DA E8 91 A7 02 41 42 7F DA E8 91 A7) |
| Key for RC/CC/DS | Corresponding to KID (recommended value: 01 23 45 67 89 AB CD EF EF CD AB 89 67 45 23 01) |

The value for Application PIN 1(Global PIN 1) shall be "0x31 0x31 0x31 0x31 0xFF 0xFF 0xFF 0xFF" and its state shall be 'disabled' during test applets execution.

The value for Application PIN 1 block shall be "0x33 0x33 0x33 0x33 0x33 0x33 0x33 0x33" and its state shall be 'disabled' during test applets execution.

Annex C (normative):  
Test file description

Every test source is written in JAVA™ and shall use methods defined in Annex D interfaces to communicate with the card, or to check status word or received data.

In order to be more readable, data specified as method string parameters shall be presented in 4 blocks of 4 bytes per line. Every block is separated by a space character. Every string line is appended to previous one and shall be aligned. An example is provided in Annex D.

Every test file shall start with a call to reset() method.

Annex D (normative):  
uicc.usim.test.util package, (U)SIM interfaces and testing script example

See attached files:

- Annex\_D\_UsimTestUtil.zip

- Annex\_D\_UsimInterfaces.zip

- Annex\_D\_Example.zip

Annex E (normative):  
Test Area files

See attached file:

- Annex\_E\_SourceCode.zip

Annex F (informative):  
Change history

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Date** | **TSG #** | **TSG Doc.** | **CR** | **Rev** | **Subject/Comment** | **New** |
| 2007-03 | CT-35 | CP-070066 | - | - |  | 2.0.0 |
| 2007-03 | CT-35 | CP-070066 | - | - |  | 6.0.0 |
| 2007-06 | - | - | - | - | Update to Rel-7 version (MCC) | 7.0.0 |
| 2008-09 | CP-41 | CP-080589 | 002 | - | Correction and completion of JavaTM files in Annex D | 7.1.0 |
| 2009-03 | - | - | - | - | Update to Rel-8 version (MCC) | 8.0.0 |
| 2009-12 | - | - | - | - | Update to Rel-9 version (MCC) | 9.0.0 |
| 2010-03 | CP-47 | CP-100180 | 005 | 1 | References update and cleanup | 9.1.0 |
| 2010-03 | CP-47 | CP-100182 | 007 | 1 | Addition of Rel-7 tests | 9.1.0 |
| 2011-03 | SP-51 | - | - | - | Update to Rel-10 version (MCC) | 10.0.0 |
| 2012-09 | SP-57 | - | - | - | Update to Rel-11 version (MCC) | 11.0.0 |
| 2013-06 | CP-60 | CP-130373 | 0009 | 1 | Improvement of results reporting | 11.1.0 |
| 2013-09 | CP-61 | CP-130529 | 0015 | 2 | Modification of the statements on security parameters | 11.2.0 |
| 2014-10 | SP-65 | - | - | - | Update to Rel-12 version (MCC) | 12.0.0 |
| 2015-12 | SP-70 | - | - | - | Update to Rel-13 version (MCC) | 13.0.0 |
| 2016-12 | CP-74 | CP-160791 | 0016 | - | Correction of CB download command coding | 13.1.0 |
| 2017-03 | -SA-75 | - | - |  | Update to Rel-14 version (MCC) | 14.0.0 |
| 2018-07 | SA-80 |  |  |  | Update to Rel-15 version (MCC) | 15.0.0 |
| 2019-12 | CP-86 | CP-193078 | 0017 | 2 | Test case 5.2.2.3, Id 22: correction of envelope content | 15.1.0 |
| 2020-07 | CP-88e | - | - | - | Update to Rel-16 version (MCC) | 16.0.0 |
| 2020-09 | CP-89e | CP-202136 | 0020 | - | Correction Test Case 5.3.6.1 Mininum security level | 16.1.0 |