ETSI TS 131 213 V18.1.0 (2025-04)

Universal Mobile Telecommunications System (UMTS);
LTE;
Test specification for (U)SIM;
Application Programming Interface (API) for Java Card™
(3GPP TS 31.213 version 18.1.0 Release 18)

Reference
RTS/TSGC-0631213vi10

Keywords
LTE,UMTS

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI IPR online database.

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™**, **LTE™** and **5G™** logo are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Legal Notice

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

The present document may refer to technical specifications or reports using their 3GPP identities. These shall be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between 3GPP and ETSI identities can be found at 3GPP to ETSI numbering cross-referencing.

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Contents

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 Card™ (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: "(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] void

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

[14] void

[15] ETSI TS 102 268 V7.3.0: "Test specification for UICC Application Programming Interface for Java Card (TM) ".

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

[17] 3GPP TS 31.101: "UICC-Terminal Interface, Physical and Logical Characteristics".

[18] 3GPP TS 31.103: "Characteristics of the IP Multimedia Services Identity Module (ISIM) application".

[19] 3GPP TS 33.220: "Generic Authentication Architecture (GAA); Generic Bootstrapping (GBA)".

[20] 3GPP TS 33.221: "Generic Authentication Architecture (GAA); Support for subscriber certificates".

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
B-TID	Bootstrapping Transaction Identifier
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
GAA	Generic Authentication Architecture
GBA	Generic Bootstrapping Architecture
GBA_ME	ME-based GBA
GBA_U	GBA with UICC-based enhancements
KDF	Key Derivation Function
Ks_int_NAF	Derived key in GBA_U which remains on UICC
NAF	Network Application Function

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

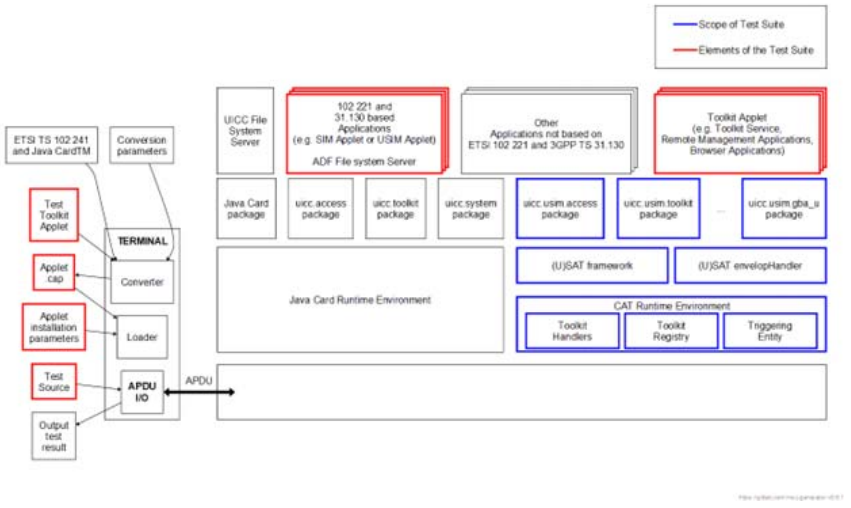
The tests defined in the present document shall be performed taking into account the services supported by the card as specified in the EF_{SST} file under DF_{GSM} as defined in TS 51.011 [5] or EF_{UST} file under ADF_{USIM} as defined in TS 31.102 [4] or EF_{IST} file under ADF_{ISIM} as defined in TS 31.103 [18].

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'

uicc.usim.gba_u package: '3'

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.

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

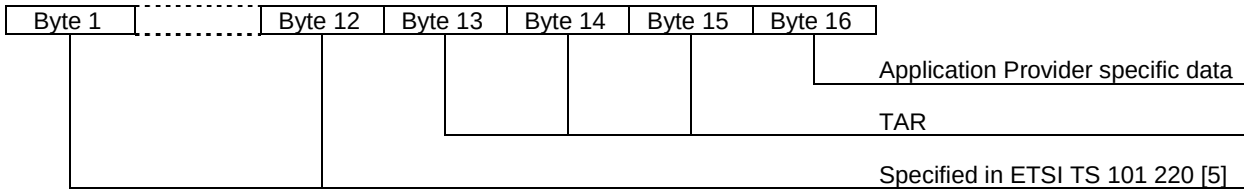
where the Test Area Reference is written in lower case.

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]*.

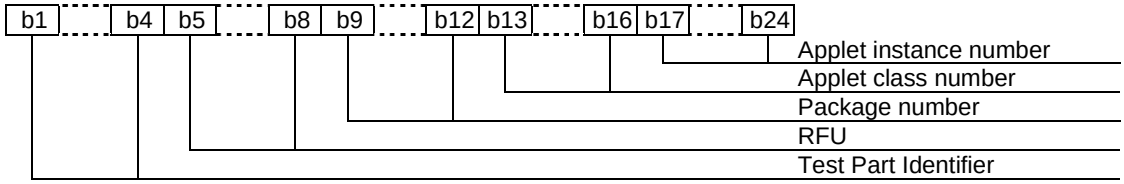
4.6 AID coding

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



TAR coding (3 bytes/ 24 bits):



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
- 0011 API uicc.usim.gba_u
- 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, uicc.usim.access and uicc.usim.gba_u 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	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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		

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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)	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
	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		

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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		

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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`
		ArrayIndexOutOfBoundsException is thrown	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
	dstLength = 1		
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	Result is 00h	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
	55 55 55 55 55		
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`
		ArrayIndexOutOfBoundsException is thrown	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
	compareOffset = -1 compareLength = 1		
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()	Result is 00h	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
	valueOffset = 2 compareOffset = 3 compareLength = 4		
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	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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.	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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:		

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
	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.	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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	Applet is triggered by an ENVELOPE (EVENT_MO_SHORT_MESSAGE_CONTROL_BY_SIM)	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
	Trigger the applet		
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	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
2	**ProactiveHandler availability with EVENT_FORMATTED_SMS_PP_UPD** 1- Update Record EF_{SMS} 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 EF_{SMS} 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		

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
	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 EF_{SMS} 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	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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 EF_{SMS} 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	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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 EF_{SMS} 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

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
4	**ProactiveResponseHandler availability with EVENT_UNFORMATTED_SMS_PP_UPD** 1- Update Record EF_{SMS} 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

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
12	**The ProactiveHandler is not available before the Terminal Profile with EVENT_FORMATTED_SMS_PP_UPD** 1- Update Record EF_{SMS} 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 EF_{SMS} 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	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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 EF_{SMS} 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.	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
4	**EnvelopeHandler availability with EVENT_UNFORMATTED_SMS_PP_UPD** 1- Update Record EF_{SMS} 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	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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 EF_{SMS} 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 EF_{SMS} 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	1- Applet1 is triggered	
	2- EnvelopeResponseHandler.getTheHandler() method is called by Applet1	2- No exception is thrown.	
	3- Applet1 builds an additional information for response packet and it calls the post() method		3- The response packet is sent
	4- Applet1 calls all methods of the EnvelopeResponseHandler (including inherited methods)	4- A ToolkitException HANDLER_NOT_AVAILABLE is thrown for each method	
	5- A EVENT_FORMATTED_SMS_PP_ENV envelope is sent to the (U)SIM	Applet1 finalizes 5- Applet1 is triggered	
	6- EnvelopeResponseHandler.getTheHandler() method is called by Applet1	6- No Exception is thrown	
	7- Applet1 builds a proactive command and it calls the send() method		7- The proactive command is sent
	8- Applet1 calls all methods of the EnvelopeResponseHandler (including inherited methods)	8- ToolkitException HANDLER_NOT_AVAILABLE is thrown for each method	

6	**EnvelopeResponseHandler availability with EVENT_UNFORMATTED_SMS_PP_ENV**		
	1- Envelope SMS-PP Download unformatted is sent to the (U)SIM	1- Applet1 is triggered	
	2- EnvelopeResponseHandler.getTheHandler() method is called by Applet1	2- No exception is thrown.	
	3- Applet1 builds the envelope response and it calls the post() method		3- The envelope response is sent
	4- Applet1 calls all methods of the EnvelopeResponseHandler (including inherited methods)	4- A ToolkitException HANDLER_NOT_AVAILABLE is thrown for each method Applet1 finalizes 5- Applet2 is triggered.	
	5- EnvelopeResponseHandler.getTheHandler() method is called	A ToolkitException HANDLER_NOT_AVAILABLE is thrown. Applet2 finalizes	
	6- An unformatted SMS PP envelope is sent to the (U)SIM	6- Applet1 is triggered.	
	7- EnvelopeResponseHandler.getTheHandler() method is called.	7- No exception is thrown.	
	8- Applet1 builds a proactive command and it calls the send() method		9- The proactive command is fetched and the Terminal response is issued.
	9- Applet1 calls all methods of the EnvelopeResponseHandler (including inherited methods)	9- A ToolkitException HANDLER_NOT_AVAILABLE is thrown for each method. Applet1 finalizes 10- Applet2 is triggered.	
	10- EnvelopeResponseHandler.getTheHandler() method is called by Applet2	A ToolkitException HANDLER_NOT_AVAILABLE is thrown.	

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	1- Applet1 is triggered	
	2- EnvelopeResponseHandler.getTheHandler() method is called by Applet1	2- No exception is thrown.	
	3- Applet1 builds an additional information for response packet and it calls the post() method		3- The response packet is sent
	4- Applet1 calls all methods of the EnvelopeResponseHandler (including inherited methods)	4- A ToolkitException HANDLER_NOT_AVAILABLE is thrown for each method	
	5- EVENT_FORMATTED_USSD envelope is sent to the (U)SIM	Applet1 finalizes 5- Applet1 is triggered	
	6- EnvelopeResponseHandler.getTheHandler() method is called by Applet1	6- No Exception is thrown	
	7- Applet1 builds a proactive command and it calls the send() method		7- The proactive command is sent
	8- Applet1 calls all methods of the EnvelopeResponseHandler (including inherited methods)	8- ToolkitException HANDLER_NOT_AVAILABLE is thrown for each method	

10	**EnvelopeResponseHandler availability with EVENT_UNFORMATTED_SMS_PP_ENV in case of multi-triggering**		
	1- Envelope USSD unformatted is sent to the (U)SIM	1- Applet1 is triggered	
	2- EnvelopeResponseHandler.getTheHandler() method is called by Applet1	2- No exception is thrown. 3- Applet1 finalizes	
		4- Applet2 is triggered.	
	5- EnvelopeResponseHandler.getTheHandler() method is called by Applet 2	5- No Exception is thrown	
	6- Applet2 calls the post() method	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 EF_{SMS} 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.	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
4	**USATEnvelopeHandler availability with EVENT_UNFORMATTED_SMS_PP_UPD** 1- Update Record EF_{SMS} 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	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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 EF_{SMS} 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 EF_{SMS} 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.

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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.

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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.

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
2	**EnvelopeHandler integrity checks with EVENT_FORMATTED_SMS_PP_UPD** 1- Update Record EF_{SMS} 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

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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. 4- 91 XX 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.	Proactive command Display Text is fetched The terminal Response of DISPLAY TEXT is sent to the (U)SIM

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
4	**EnvelopeHandler integrity checks with EVENT_UNFORMATTED_SMS_PP_UPD** 1- Update Record EF_{SMS} instruction single and unformatted is sent to the (U)SIM	1- Applet is triggered	
	2- EnvelopeHandler.getTheHandler() method is called	2- No exception is thrown.	
	3- Copy the content of the EnvelopeHandler in buffer1 using EnvelopeHandler.copy()	3- No exception is thrown.	
	The EnvelopeHandler.findTLV method is called with TAG_SMS_TPDU		4- 91 XX
	4- A proactive command DISPLAY TEXT is sent 5- Envelope call control by SIM is sent to the (U)SIM	5- Applet is triggered	
	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	6- No exception is thrown and the handler contains the envelope call control by SIM	
	The EnvelopeHandler.findTLV() method is called with TAG_DEVICE_IDENTITIES Call Control execution is finished.		Proactive command Display Text is fetched The terminal Response of DISPLAY TEXT is sent to the (U)SIM
	Check that the SMS_TPDU is the TLV selected 7- The content of EnvelopeHandler is compared with buffer1 using Util.arrayCompare()	7- The contents of the EnvelopeHandler shall be the same as stored in buffer1.	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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. 4- 91 XX 5- Applet is triggered 6- No exception is thrown and the handler contains the envelope call control by SIM Proactive command Display Text is fetched The terminal Response of DISPLAY TEXT is sent to the (U)SIM 7- The contents of the EnvelopeHandler shall be the same as stored in buffer1.	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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. 4- 91 XX 5- Applet is triggered 6- No exception is thrown and the handler contains the envelope call control by SIM Proactive command Display Text is fetched The terminal Response of DISPLAY TEXT is sent to the (U)SIM 7- The contents of the EnvelopeHandler shall be the same as stored in buffer1.	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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. 4- 91 XX 5- Applet is triggered 6- No exception is thrown and the handler contains the envelope call control by SIM Proactive command Display Text is fetched The terminal Response of DISPLAY TEXT is sent to the (U)SIM 7- The contents of the EnvelopeHandler shall be the same as stored in buffer1.	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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 EF_{SMS} APDU into a COMPREHENSION TLV List containing Device Identities TLV, Address TLV, SMS TPDU TLV and AID TLV (only if the EF_{SMS} 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 EF_{SMS} file and the value of the EF_{SMS} 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

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

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
2	**USATEnvelopeHandler integrity checks with EVENT_FORMATTED_SMS_PP_UPD** 1- Update Record EF_{SMS} 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

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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. 4- 91 XX 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.	Proactive command Display Text is fetched The terminal Response of DISPLAY TEXT is sent to the (U)SIM

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
4	**USATEnvelopeHandler integrity checks with EVENT_UNFORMATTED_SMS_PP_UPD** 1- Update Record EF_{SMS} instruction single and unformatted is sent to the (U)SIM	1- Applet is triggered	
	2- USATEnvelopeHandler.getTheHandler() method is called	2- No exception is thrown.	
	3- Copy the content of the EnvelopeHandler in buffer1 using USATEnvelopeHandler.copy()	3- No exception is thrown.	
	The USATEnvelopeHandler.findTLV method is called with TAG_SMS_TPDU		4- 91 XX
	4- A proactive command DISPLAY TEXT is sent	5- Applet is triggered	
	5- Envelope call control by SIM is sent to the (U)SIM		
	USATEnvelopeHandler.getTheHandler() method is called	6- No exception is thrown and the handler contains the envelope call control by SIM	
	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.		Proactive command Display Text is fetched The terminal Response of DISPLAY TEXT is sent to the (U)SIM
	Check that the TAG_DEVICE_IDENTITIES is the TLV selected		
	7- The content of USATEnvelopeHandler is compared with buffer1 using Util.arrayCompare()	7- The contents of the EnvelopeHandler shall be the same as stored in buffer1.	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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. 4- 91 XX 5- Applet is triggered 6- No exception is thrown and the handler contains the envelope call control by SIM Proactive command Display Text is fetched The terminal Response of DISPLAY TEXT is sent to the (U)SIM 7- The contents of the EnvelopeHandler shall be the same as stored in buffer1.	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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. 4- 91 XX 5- Applet is triggered 6- No exception is thrown and the handler contains the envelope call control by SIM Proactive command Display Text is fetched The terminal Response of DISPLAY TEXT is sent to the (U)SIM 7- The contents of the EnvelopeHandler shall be the same as stored in buffer1.	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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.	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
9	**getLength() call** `Call getLength() method`	return the Comprehension TLV list length	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
10	**getEnvelopeTag() call** `Call getTag() method`	return BTAG_SMS_PP_DOWNLOAD	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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.	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
12	**getLength() call** `Call getLength() method`	return the Comprehension TLV list length	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
13	**getEnvelopeTag() call** `Call getEnvelopeTag() method`	return BTAG_SMS_PP_DOWNLOAD	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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. 4- 91 XX 5- Applet is triggered 6- No exception is thrown and the handler contains the envelope call control by SIM Proactive command Display Text is fetched The terminal Response of DISPLAY TEXT is sent to the (U)SIM 7- The contents of the EnvelopeHandler shall be the same as stored in buffer1.	

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

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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)	1- Applet1 is triggered, sends a ISOException with the reason code REPLY_BUSY then finalizes	1- SW = 93 00
	2- Send an envelope Event Unformatted SMS CB (multi triggering event, single registered applet)	2- Applet2 is triggered, sends a ISOException with the reason code REPLY_BUSY then finalizes	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)	1- Applet1 is triggered, sends a ISOException with reason code different to REPLY_BUSY then finalizes	1- SW = 90 00
	2- Send an envelope Event formatted SMS PP ENV to trigger Applet1 (single triggering event)	2- Applet1 is triggered, sends a ToolkitException then finalizes	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)	1- Applet1 is triggered, sends a ISOException with the reason code REPLY_BUSY then finalizes	1- SW = 93 00
	2- Send an envelope Event unformatted SMS CB (multi triggering event, single registered applet)	2- Applet2 is triggered, sends a ISOException with the reason code REPLY_BUSY then finalizes	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.

5.3.4.1.1.4 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	

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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 EF_{SMS} 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 EF_{SMS} APDU. 3. Short Message Point to Point Concatenated Formatted is received by Update Record EF_{SMS} 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 EF_{SMS} APDU. 2. Short Message Point to Point Concatenated and Formatted is received by Update Record EF_{SMS} 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 EF_{SMS} APDU. 4. Short Message Point to Point Concatenated Formatted is received by Update Record EF_{SMS} 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 EF_{SMS} 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 EF_{SMS} APDU 3. Short Message Point to Point Concatenated and Unformatted is received by Update Record EF_{SMS} 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 EF_{SMS} APDU 2. Short Message Point to Point Concatenated and Unformatted is received by Update Record EF_{SMS} 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 EF_{SMS} APDU 4. Short Message Point to Point Concatenated and Unformatted is received by Update Record EF_{SMS} 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`		

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
	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 Void

5.3.4.8 Void

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.

5.3.5.1.2 Test area files

Test Source: Test_Ufw_Erp_Emcn.java

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.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.

5.3.6.2.1.2 Parameter errors

No requirements.

5.3.6.2.1.3 Context errors

No requirements.

5.3.6.2.2 Test area files

Test source: Test_Ufw_Tin_Tar.java

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

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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.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 EF_{SMS} 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.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

CRR Number	Test Case Number
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

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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 EF_{SMS} 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 EF_{SMS} 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 EF_{SMS} instruction with status words 9000 2- The USAT Framework answers to the Update Record EF_{SMS} instruction with status words 9000

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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 EF_{SMS} 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 EF_{SMS} 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 EF_{SMS} 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 EF_{SMS} 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 EF_{SMS} instruction with status words 9000 2- The USAT Framework answers to the Update Record EF_{SMS} instruction with status words 9000 3- The USAT Framework answers to the Update Record EF_{SMS} instruction with status words 9000 4- The USAT Framework answers to the Update Record EF_{SMS} instruction with status words 9000
9	**Update Record EF_{SMS} instruction formatted with wrong cryptographic checksum** 1- Short Message single and formatted is sent to the USAT Framework by Update Record EF_{SMS} 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 EF_{SMS} 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 EF_{SMS} instruction with status words 9000 2- The USAT Framework answers to the Update Record EF_{SMS} instruction with status words 9000

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
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.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	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.

Id	Description	API/(U)SAT Framework Expectation	APDU Expectation
	Data in plain text = "APPLET1"		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 Void

5.4 Package uicc.usim.gba_u package

5.4.1 Class GBAUCipher

5.4.1.1 Method getInstance (byte, boolean)

Test Area Reference: Api_3_Gci_Gin1

5.4.1.1.1 Conformance requirements

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

```
public static GBAUCipher getInstance(byte algorithm,
                                     boolean externalAccess)
                              throws javacard.security.CryptoException
```

5.4.4.1.1.1 Normal execution

- CRRN1: The method shall return the single system instance of the GBAUCipher class.
- CRRN2: The GBAUCipher is a Temporary JCRE Entry Point Object.

5.4.4.1.1.2 Parameter Errors

- CRRP1: The method throws javacard.security.CryptoException with reason CryptoException.NO_SUCH_ALGORITHM when the requested algorithm is an asymmetric algorithm.

5.4.4.1.1.3 Context errors

No requirements.

5.4.1.1.2 Test area files

Test Source: Test_Api_3_Gci_Gin.java

Test Applet: Api_3_Gci_Gin1_1.java

Api_3_Gci_Gin2_1.java

Cap File: Api_3_Gci_Gin.cap

5.4.1.1.3 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.
P1	4

5.4.1.1.4 Test procedure

Id	Description	API Expectation	APDU Expectation
1	Call getInstance() method twice	The returned objects shall be the same algorithm parameter is set to javacardx.crypto.Cipher.CIPHER_AES_CBC externalAccessparameter is set to false	
2	Verify that getInstance() method returns an GBAUCipher class	The reference returned shall be an object instance of GBAUCipher class (check cast) algorithm parameter is set to javacardx.crypto.Cipher.CIPHER_AES_CBC externalAccessparameter is set to false	
3	Verify the returned value is not null	The reference returned shall not be null. algorithm parameter is set to javacardx.crypto.Cipher.CIPHERAES_CBC externalAccessparameter is set to false	
4	Verify that getInstance() throws javacard.security.CryptoException with reason CryptoException.NO_SUCH_ALGORITHM	The reference returned shall not be null. algorithm parameter is set to javacardx.crypto.Cipher.CIPHER_RSA externalAccessparameter is set to false	

5.4.2.1 Method doFinal(byte[] inBuff, short inOffset, short inLength, byte[] outBuff, short outOffset)

Test Area Reference: Api_3_Gci_Dfin

5.4.2.1.1 Conformance requirements

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

```
public abstract short doFinal(byte[] inBuff,
                              short inOffset,
                              short inLength,
                              byte[] outBuff,
                              short outOffset)
                       throws javacard.security.CryptoException
```

5.4.2.1.1.1 Normal execution

- CRRN1: The method shall return encrypted/decrypted output from all/last input data using Ks_int_NAF linked to NAF ID given in init()..
- CRRN2: This method shall process any remaining input data buffered by one or more calls to the update() method as well as input data supplied in the inBuff parameter.
- CRRN3: After a call to this method, this Cipher object shall be reset to the state it was in when previously initialized via a call to init() methods. The initial vector(IV) used in AES, DES, Korean SEED and SM4 algorithms will be reset to 0.
- CRRN4: When using block-aligned data (multiple of block size), if the input buffer, inBuff and the output buffer, outBuff are the same array, then the output data area must not partially overlap the input data area such that the input data is modified before it is used; if inBuff==outBuff and inOffset < outOffset < inOffset+inLength, incorrect output may result.

5.4.2.1.1.2 Parameter errors

- CRRP1: The method throws java.lang.NullPointerException if inBuff or outOffset is null.
- CRRP2: The method throws java.lang.ArrayIndexOutOfBoundsException if the check operation on inBuff or outOffset would cause access of data outside array bounds.

5.4.2.1.1.3 Context errors

- CRRC1: The method throws javacard.security.CryptoException with the reason code CryptoException.UNINITIALIZED_KEY if the key (Ks_int_NAF key) is uninitialized.

- CRRC2: The method throws javacard.security.CryptoException with the reason code CryptoException.INVALID_INIT if this GBAUCipher object is not initialized.

- CRRC3: The method throws javacard.security.CryptoException with the reason code CryptoException.ILLEGAL_USE if this GBAUCipher algorithm does not pad the message and the message is not block aligned.

- CRRC4: The method throws javacard.security.CryptoException with the reason code CryptoException.ILLEGAL_USE if this GBAUCipher algorithm does not pad the message and no input data has been provided in inBuff or via the update() method.

- CRRC5: The method throws javacard.security.CryptoException with the reason code CryptoException.ILLEGAL_USE if the decrypted data is not bounded by appropriate padding bytes.

5.4.2.1.2 Test area files

Test Source: Test_Api_3_Gci_Dfin.java

Testcase_Api_3_Gci_Dfin_1.java

Test Applet: Api_3_Gci_Dfin_1.java

Cap File: Api_3_Gci_Dfin_1.cap

5.4.2.1.3 Test coverage

CRR number	Test case number
N1	10
N2	12
N3	10
N4	11
P1	1,2
P2	3,4
C1	5
C2	6
C3	7
C4	8
C5	9

5.4.2.1.4 Test procedure

Id	Description	API Expectation	APDU Expectation
0	The content of EF for Access Control to GBA_U_APIs include NAF_ID1 and the AID of app1, which means NAF_ID1 is corresponding to app1.		
	Parameter errors		
1	1- Execute GBA_U Procedure with NAF_ID1 and select app1. 2- Create the NAF key and set the key value. 3- Create a GBAUCipher object instance. 4- Call init() method with theMode of MODE_ENCRYPT. 5- Call doFinal() method, the inBuff is set to null, and the other parameters are correct. 6- Call init() method with theMode of MODE_DECRYPT. 7- Call doFinal() method, the inBuff is set to null, and the other parameters are correct..	5- Throw java.lang.NullPointerException 7- Throw java.lang.NullPointerException	

2	1- Execute GBA_U Procedure with NAF_ID1 and select app1. 2- Create the NAF key and set the key value. 3- Create a GBAUCipher object instance. 4- Call init() method with theMode of MODE_ENCRYPT. 5- Call doFinal() method, the outOffset is set to null, and the other parameters are correct. 6- Call init() method with theMode of MODE_DECRYPT. 7- Call doFinal() method, the outOffset is set to null, and the other parameters are correct.	5- Throw java.lang.NullPointerException 7- Throw java.lang.NullPointerException	
3	1- Execute GBA_U Procedure with NAF_ID1 and select app1. 2- Create the NAF key and set the key value. 3- Create a GBAUCipher object instance. 4- Call init() method with theMode of MODE_ENCRYPT. 5- Call doFinal() method, inOffset is equal to inbuff.length, and the other parameters are correct. 6- Call doFinal() method,inOffset is equal to -1, and the other parameters correct. 7- Call doFinal() method,inLength is equal to inbuff.length+1, and the other parameters are correct. 8- Call doFinal() method, inOffset is equal to 1, inLength is equal to inbuff.length, and the other parameters correct. 9- Call init() method with theMode of MODE_DECRYPT. 10- Call doFinal() method, inOffset is equal to the length of inbuff, and the other parameters are correct. 11- Call doFinal() method, inOffset is equal to -1, and other parameters are correct. 12- Call doFinal() method, inLength is equal to inbuff.length+1, and the other parameters are correct. 13- Call doFinal() method, inOffset is equal to 1, inLength is equal to inbuff.length, and the other parameters are correct.	Step 5 to 8 - Throw java.lang.ArrayIndexOutOfBoundsException Step 11 to 13 - Throw java.lang.ArrayIndexOutOfBoundsException	

4	1- Execute GBA_U Procedure with NAF_ID1 and select app1. 2- Create the NAF key and set the key value. 3- Create a GBAUCipher object instance. 4- Call init() method with theMode of MODE_ENCRYPT. 5- Call doFinal() method, outOffset is equal to outbuff.length, and the other parameters are correct. 6- Call doFinal() method, outOffset is equal to -1, and the other parameters are correct. 7- Call doFinal() method, the length of the return value is equal to outbuff.length+1, and the other parameters are correct. 8- Call doFinal() method, outOffset is equal to 1, the length of the return value is equal to outbuff.length, and the other parameters are correct. 9- Call init() method with theMode of MODE_DECRYPT. 10- Call doFinal() method, the outOffset is equal to outbuff.length, and the other parameters are correct. 11- Call doFinal() method, the outOffset is equal to -1, and the other parameters are correct. 12- Call doFinal() method, the length of the return value is equal to outbuff.length+1, and the other parameters are correct. 13- Call doFinal() method, the outOffset is equal to 1, the length of the return value is equal to inbuff.length, and the other parameters are correct.	Step 5 to 8 - Throw java.lang.ArrayIndexOutOfBoundsException Step 11 to 13 - Throw java.lang.ArrayIndexOutOfBoundsException	
	Context errors		
5	1- Execute GBA_U Procedure with NAF_ID1 and select app1. 2- Create the NAFkey and set the key value. 3- Create a GBAUCipher object instance. 4- Call init() method with theMode of MODE_ENCRYPT. 5- Clear the NAFkey value. 6- Call doFinal() method. 7- Set the NAFkey value. 8- Call init() method with theMode of MODE_DECRYPT. 9- Call doFinal() method.	6 - Throw javacard.security.CryptoException with the reason code CryptoException.UNINITIALIZED_KEY 9 - Throw javacard.security.CryptoException with the reason code CryptoException.UNINITIALIZED_KEY	
6	1- Execute GBA_U Procedure with NAF_ID1 and select app1. 2- Create a GBAUCipher object instance. 3- Call doFinal() method. 4- Call init() method to initialize the GBAUCipher object with the key object with theMode value 0xFF. 5- Call doFinal() method.	3-Throw javacard.security.CryptoException with the reason code CryptoException.INVALID_INIT. 5- Throw javacard.security.CryptoException with the reason code CryptoException.INVALID_INIT	

7	1- Execute GBA_U Procedure with NAF_ID1 and select app1. 2- Create the NAF key and set the key value. 3- Create a GBAUCipher object instance. 4- Call init() method to initialize GBAUCipher object with key object, and theMode value is set to MODE_ENCRYPT. 5- Call doFinal() method, and the value of inLength is set to10.	5- Throw javacard.security.CryptoException with the reason code CryptoException.ILLEGAL_USE	
8	1- Execute GBA_U Procedure with NAF_ID1 and select app1. 2- Create the NAF key and set the key value. 3- Create a GBAUCipher object instance. 4- Call init() method to initialize GBAUCipher object with key object, and theMode value is set to MODE_ENCRYPT. 5- Call doFinal() method, and the value of inLength is set to 0. 6- Call updata() method, and the value of inLength is set to 0. 7- Call doFinal() method, and the value of inLength is set to 0.	5- Throw javacard.security.CryptoException with the reason code CryptoException.ILLEGAL_USE 7- Throw javacard.security.CryptoException with the reason code CryptoException.ILLEGAL_USE	
9	1- Execute GBA_U Procedure with NAF_ID1 and select app1. 2- Create the NAF key and set the key value. 3- Create a GBAUCipher object instance. 4- Call init() method to initialize GBAUCipher object with key object, and theMode value is set to MODE_DECRYPT. 5- Call doFinal() method, and the value of inLength is equal to the value of the correct length minus one. 6- Call doFinal() method, and the value of inLength is equal to the value of the correct length plus one. 7-Call doFinal() method, and the value of inLength is set to 0.	5- Throw javacard.security.CryptoException with the reason code CryptoException.ILLEGAL_USE 6- Throw javacard.security.CryptoException with the reason code CryptoException.ILLEGAL_USE 7- Throw javacard.security.CryptoException with the reason code CryptoException.ILLEGAL_USE	
	Normal execution		
10	Note:The test of symmetric algorithm supports all key types, and when the algorithm supports padding, the value of input plaintext n is 1, and when the algorithm does not support padding, the block size of the input plaintext is n.	4- The return value of dofinal is n, and the decryption result is consistent with the input plaintext. 5- Two return values of dofinal() method should be correct. 6- Two return values of dofinal() method should be same and correct.	

	1- Execute GBA_U Procedure with NAF_ID1 and select app1. 2- Create the NAF key and set the key value. 3- Create a GBAUCipher object instance. 4- Call init() method with theMode value of MODE_ENCRYPT,call dofinal() method to encrypt n-byte plaintext. Call init() method with theMode value of MODE_DECRYPT, call dofinal method to decrypt the encryption datas. 5- Call init() method with theMode value of MODE_ENCRYPT, call dofinal() method to encrypt n bytes plaintext, Call init() method with theMode value of MODE_ENCRYPT, call dofinal() method to encrypt n+6 bytes plaintext. 6- Call init() method with theMode value of MODE_ENCRYPT, call dofinal() method to encrypt n byte plaintext, call init() method with theMode value of MODE_ENCRYPT, call dofinal() method to encrypt the same n byte plaintext.		
11	1- Execute GBA_U Procedure with NAF_ID1 and select app1. 2- Create the NAF key and set the key value. 3- Create a GBAUCipher object instance. 4- Call init() method with theMode value of MODE_ENCRYPT(using block-aligned data), call dofinal() method to encrypt 8 bytes plaintext, inoffset=5, outoffset=1, inbuff=outbuff, call init() method with theMode value of MODE_DECRYPT, call dofinal() method to decrypt the encryption result, inoffset=5, outoffset=4, inbuff=outbuff. 5- Call init() method with theMode value of MODE_ENCRYPT(using block-aligned data), call dofinal() method to encrypt 8 bytes plaintext, inoffset=1, outoffset=18, inbuff=outbuff, call init() method with theMode value of MODE_DECRYPT, call dofinal() method to decrypt the encrypted result, inoffset=1, outoffset=18, inbuff=outbuff	4- The decryption result of dofinal() method should be consistent with the plaintext. 5- The decryption result of dofinal() method should be consistent with the plaintext.	
12	1- Execute GBA_U Procedure with NAF_ID1 and select app1. 2- Create the NAF key and set the key value. 3- Create a GBAUCipher object instance. 4- Call init() method with theMode value of MODE_ENCRYPT, call update() method to accumulate 5 bytes of plaintext, call dofinal() method to encrypt n bytes of plaintext. 5- Call init() method with theMode value of MODE_DECRYPT, call update() method to accumulate 5 bytes of ciphertext, call dofinal() method to decrypt the encryption result.	4- The return of dofinal() method should be correct. 5- The return value' length of dofinal() method is n+5, and the decryption result should be consistent with the plaintext.	

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

A.3 Acronyms for GBAU_U API part (uicc.usim.gba_u)

`GBAUCipher`	Gci
`GBAUSignature`	Gsi

A.3.1 GBAUCipher method

Method name	Acronyms
`GBAUCipher getInstance(byte,boolean)`	Gin1
`GBAUCipher getInstance(byte,byte,boolean)`	Gin2
`void init(byte,byte[],short,short,byte[],short,short)`	Ini1

void init(byte,byte[],short,short,byte[],short,short, byte[],short,short)	Ini2
short update(byte[],short,short,byte[],short)	Updt
short doFinal(byte[],short,short,byte[],short)	Dfin
byte getAlgorithm()	Galg
byte getCipherAlgorithm()	Gcal
byte getPaddingAlgorithm()	Gpal

A.3.2 GBAUSignature method

Method name	Acronyms
GBAUSignature getInstance(byte,boolean)	Gin1
GBAUSignature getInstance(byte,byte,boolean)	Gin2
void init(byte,byte[],short,short,byte[],short,short)	Ini1
void init(byte,byte[],short,short,byte[],short,short, byte[],short,short)	Ini2
short update(byte[],short,short)	Updt
short sign(byte[],short,short,byte[],short)	Sign
boolean verify(byte[],short,short,byte[],short,short)	Veri
byte getAlgorithm()	Galg
byte getCipherAlgorithm()	Gcal
short getLength()	Glen
byte getMessageDigestAlgorithm()	Gmda
byte getPaddingAlgorithm()	Gpal
void setInitialDigest(byte[],short,short,byte[],short,short)	Sidi
short signPreComputedHash(byte[],short,short,byte[],short)	Spch
boolean verifyPreComputedHash(byte[],short,short,byte[],short, short)	Vpch

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 EF_{SMS}.

B.1 Under MF and $DF_{TELECOM}$

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

Name	Identifier	Description	Special Notes
EF_{DIR}	2F00	Record 1: 61 12 4F 10 AID_1 Record 2: 61 12 4F 10 AID_2 Record 3: 61 yy 4F xx ADF USIM AID	Where: - xx is the length of ADF_{USIM} AID - yy = xx + 2
EF_{SMS}	6F3C	1st record: 00 FF … FF(length 176) 2nd record: 00 FF … FF(length 176) 3rd record: 00 FF … FF(length 176)	The file EF_{SMS} should be present under both MF/$DF_{TELECOM}$ and directly under ADF1. One of these two EF_{SMS} 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.

B.2 ADF USIM

B.2.1 ADF USIM Files

This clause contains a figure depicting the file structure the ADF_{USIM}. ADF_{USIM} shall be selected using the AID and information in EF_{DIR}. The USIM AID is defined in the Annex O of 3GPP TS 31.101 [17].

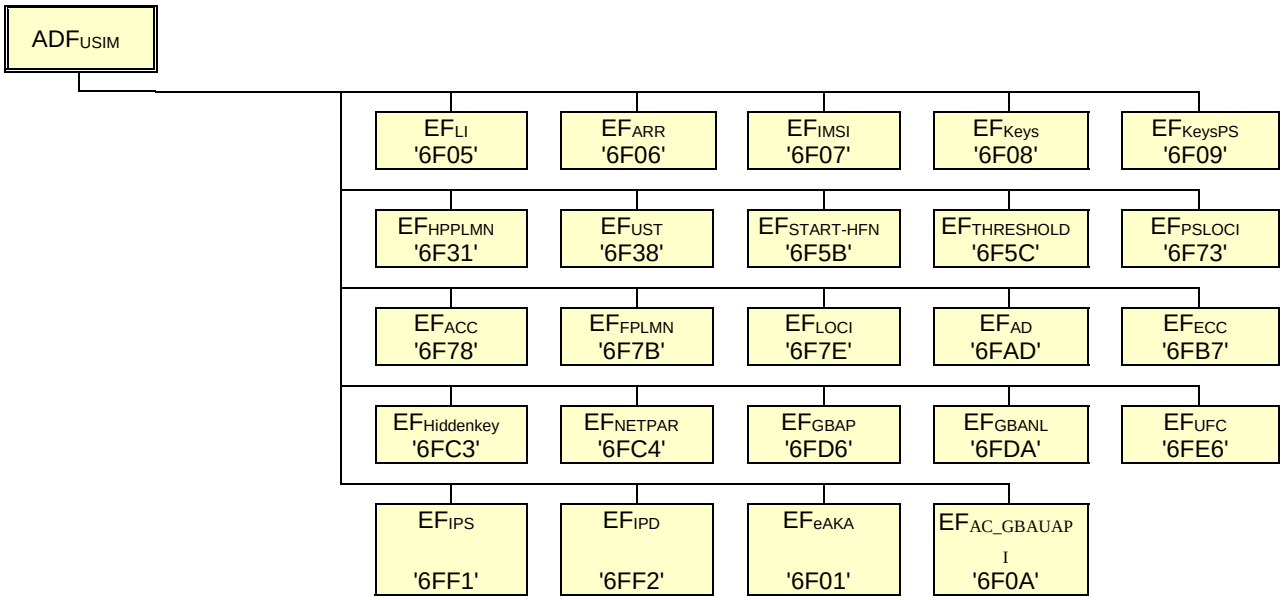


Figure B.2.1: File identifiers and directory structures of USIM

B.2.2 EF_{LI} (Language Indication)

As defined in TS 31.102 [4] clause 4.2.1 for the file detail and Annex E for the file content.

B.2.3 EF_{ARR} (Access Rule Reference)

As defined in TS 31.102 [4] clause 4.2.55 for the file detail and Annex E for the file content.

B.2.4 EF_{IMSI} (IMSI)

As defined in TS 31.102 [4] clause 4.2.2 for the file detail.

The file content is as follow:

Logically: 2460813579

Coding:	B1	B2	B3	B4	B5	B6	B7	B8	B9
Hex	06	21	64	80	31	75	F9	FF	FF

B.2.5 EF_{Keys} (Ciphering and Integrity Keys)

As defined in TS 31.102 [4] clause 4.2.3 for the file detail and Annex E for the file content.

B.2.6 EF_{KeysPS} (Ciphering and Integrity Keys for Packet Switched domain)

As defined in TS 31.102 [4] clause 4.2.4 for the file detail and Annex E for the file content.

B.2.7 EF_{HPPLMN} (Higher Priority PLMN search period)

As defined in TS 31.102 [4] clause 4.2.6 for the file detail and Annex E for the file content.

B.2.8 EF_{UST} (USIM Service Table)

As defined in TS 31.102 [4] clause 4.2.8 for the file detail.

The file content is as follow:

Contents:

Service n°68	Generic Bootstrapping Architecture (GBA)	available
Service n°33	shall be set to '1'	available

All other services shall be set to "not available".

B.2.9 $EF_{START-HFN}$ (Initialisation values for Hyperframe number)

As defined in TS 31.102 [4] clause 4.2.51 for the file detail and Annex E for the file content.

B.2.10 $EF_{THRESHOLD}$ (Maximum value of START)

As defined in TS 31.102 [4] clause 4.2.52 for the file detail and the file content set to 'F1 00 00':

B.2.11 EF_{PSLOCI} (Packet Switched location information)

As defined in TS 31.102 [4] clause 4.2.23 for the file detail and Annex E for the file content.

B.2.12 EF_{ACC} (Access Control Class)

As defined in TS 31.102 [4] clause 4.2.15 for the file detail.

The file content is as follow:

Logically: One and one access class from 0 – 9, e.g. class 7 for which the coding is "00 80".

B.2.13 EF_{FPMLN} (Forbidden PLMNs)

As defined in TS 31.102 [4] clause 4.2.16 for the file detail and Annex E for the file content.

B.2.14 EF_{LOCI} (Location Information)

As defined in TS 31.102 [4] clause 4.2.17 for the file detail and Annex E for the file content.

B.2.15 EF_{AD} (Administrative Data)

As defined in TS 31.102 [4] clause 4.2.18 for the file detail.

The file content is as follow:

Logically:

Mode of operation:	normal operation
Additional information:	ciphering indicator feature disabled
Length of MNC in the IMSI:	3 digit

Coding:	B1	B2	B3	B4
Hex	00	00	00	03

B.2.16 EF_{ECC} (Emergency Call Codes)

As defined in TS 31.102 [4] clause 4.2.21 for the file detail.

The file content is as follow:

Logically: Emergency call code: "122";
Emergency call code alpha identifier: "TEST";
Emergency call Service Category: Mountain Rescue.

Coding:	B1	B2	B3	B4	B5	B6	B7	B8
Hex	21	F2	FF	54	45	53	54	10

B.2.17 $EF_{Hiddenkey}$ (Key for hidden phone book entries)

As defined in TS 31.102 [4] clause 4.2.42 for the file detail and Annex E for the file content.

B.2.18 EF_{NETPAR} (Network Parameters)

As defined in TS 31.102 [4] clause 4.2.57 for the file detail and Annex E for the file content.

B.2.19 EF_{GBABP} (GBA Bootstrapping parameters)

As defined in TS 31.102 [4] clause 4.2.79 for the file detail and Annex E for the file content.

B.2.20 EF_{GBANL} (GBA NAF List)

As defined in TS 31.102 [4] clause 4.2.83 for the file detail and Annex E for the file content.

B.2.21 EF_{UFC} (USAT Facility Control)

As defined in TS 31.102 [4] clause 4.2.93 for the file detail and Annex E for the file content.

B.2.22 EF_{IPS} (IMEI(SV) Pairing Status)

As defined in TS 31.102 [4] clause 4.2.101 for the file detail and Annex E for the file content.

B.2.23 EF_{IPD} (IMEI(SV) of Pairing Device)

As defined in TS 31.102 [4] clause 4.2.102 for the file detail and Annex E for the file content.

B.2.24 EF_{eAKA} (enhanced AKA support)

As defined in TS 31.102 [4] clause 4.2.1 for the file detail and the file content set to '00' (enhanced SQN calculation not supported).

B.2.25 $EF_{AC\_GBAUAPI}$ (Access Control to GBA_U_API)

As defined in TS 31.102 [4] clause 4.2.116 for the file detail.

The file content is as follow:

Record 1:

Logically:

Applet Test_api_3_Gci_Gin1: AID A0000000 871005FF FFFFFF89 30010102

NAF ID : FQDN of the NAF, concatenated with the Ua security protocol identifier, with

FQDN: "naf01"

Ua security protocol identifier: 0x01,0x00,0x00,0x00,0x00 (3GPP Ua security protocol according to TS 33.221 [TS33.221])

Coding:

B1	B2	B3	B4	B5	B6	B7	B8	B9	B10
80	1D	10	A0	00	00	00	87	10	05
B11	B12	B13	B14	B15	B16	B17	B18	B19	B20
FF	FF	FF	FF	89	30	01	01	02	0B
B21	B22	B23	B24	B25	B26	B27	B28	B29	B30
05	6E	61	66	30	31	01	00	00	00
B31									
00									

B.2.26 EF_{LI} (Language Indication)

As defined in TS 31.102 [4] clause 4.2.1 for the file detail and Annex E for the file content.

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

NOTE: Since version 18.1.0 the export files are delivered in export format version 2.3.

Annex E (normative): Test Area files

See attached file:

- Annex_E_SourceCode.zip

NOTE: Since version 18.1.0 the Ant build file is available in root of "Annex_E_SourceCode.zip" to facilitate generation of attached files from Annex C, D and E.

Annex F (informative): Change history

Change history							
Date	**Meeting**	**TDoc**	**CR**	**Rev**	**Cat**	**Subject/Comment**	**New version**
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 Minimum security level	16.1.0
2022-04	-	-	-	-	-	Update to Rel-17 version (MCC)	17.0.0
2024-03	-	-	-	-	-	Update to Rel-18 version (MCC)	18.0.0
2025-03	CT#107	CP-250062	0024	-	F	Corrections on document vs code discrepancy and cleaning on test case without source code	18.1.0
2025-03	CT#107	CP-250062	0025	-	F	Corrections on document test cases without source code	18.1.0
2025-03	CT#107	CP-250063	0022	5	B	GBA_U API test case	18.1.0

History

Document history		
V18.0.0	May 2024	Publication
V18.1.0	April 2025	Publication