3GPP TSG-T plenary meeting #20
Hämeenlinna, Finland, 4-6 May 2003

Tdoc TP-030125

Source: T3

Title: CRs to TS 51.013: Test specification for the SIM API for Java Card^{TM}

Document for: Approval

This document contains the following change request:

T3 Doc	Spec	CR	Rev	Phase	Subject	Cat	V. old	V. new
T3-030409	51.013	001	-	Rel-5	Update of 51.013 Specification for Release 5	B	4.0.1	5.0.0

3GPP TSG-T3 Meeting #27
Sapporo, Japan, 20-23 May, 2003

Tdoc T3-030409
Rev T3-030300

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] (void)

[2] (void)

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

[4] 3GPP TS 11.14~~51.014~~: "
Specification of the SIM application toolkit for the Subscriber Identity Module – Mobile Equipment (SIM – ME) interface".

[5] 3GPP TS~~GSM~~ 11.17: "Subscriber Identity Module" (SIM) conformance test specification".

[6] (void)

[7] 3GPP TS ~~GSM~~ 43.019 Rel-5: " Subscriber Identity Module Application Programming Interface (SIM API)~~; SIM API~~ for Java Card™; Stage 2".

[8] 3GPP TS 23.048 Rel-~~4~~5: " Security Mechanisms for the (U)SIM application toolkit; Stage 2"

[9] ISO/IEC 7816-3 (1997) " Identification cards - Integrated circuit(s) cards with contacts, Part 3: Electronic signals and transmission protocols".

[10] 3GPP TS 42.019: " Subscriber Identity Module Application Programming Interface (SIM API); Service description; Stage 1".

[11] SUN Java Card Specification "Java Card 2.1 API Specification".

[12] SUN Java Card Specification "Java Card 2.1 Runtime Environment Specification".

[13] SUN Java Card Specification "Java Card 2.1 VM Architecture Specification".

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

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

[15] 3GPP TS~~GSM~~ 11.10-1: "Digital cellular telecommunication system (Phase 2+); Mobile Station (MS) conformance specification; Part 1: Conformance specification".

4 Test Environment

4.1 Applicability

The tests defined in this specification shall be performed taking into account the services supported by the card as specified in the EF_{SST} file.

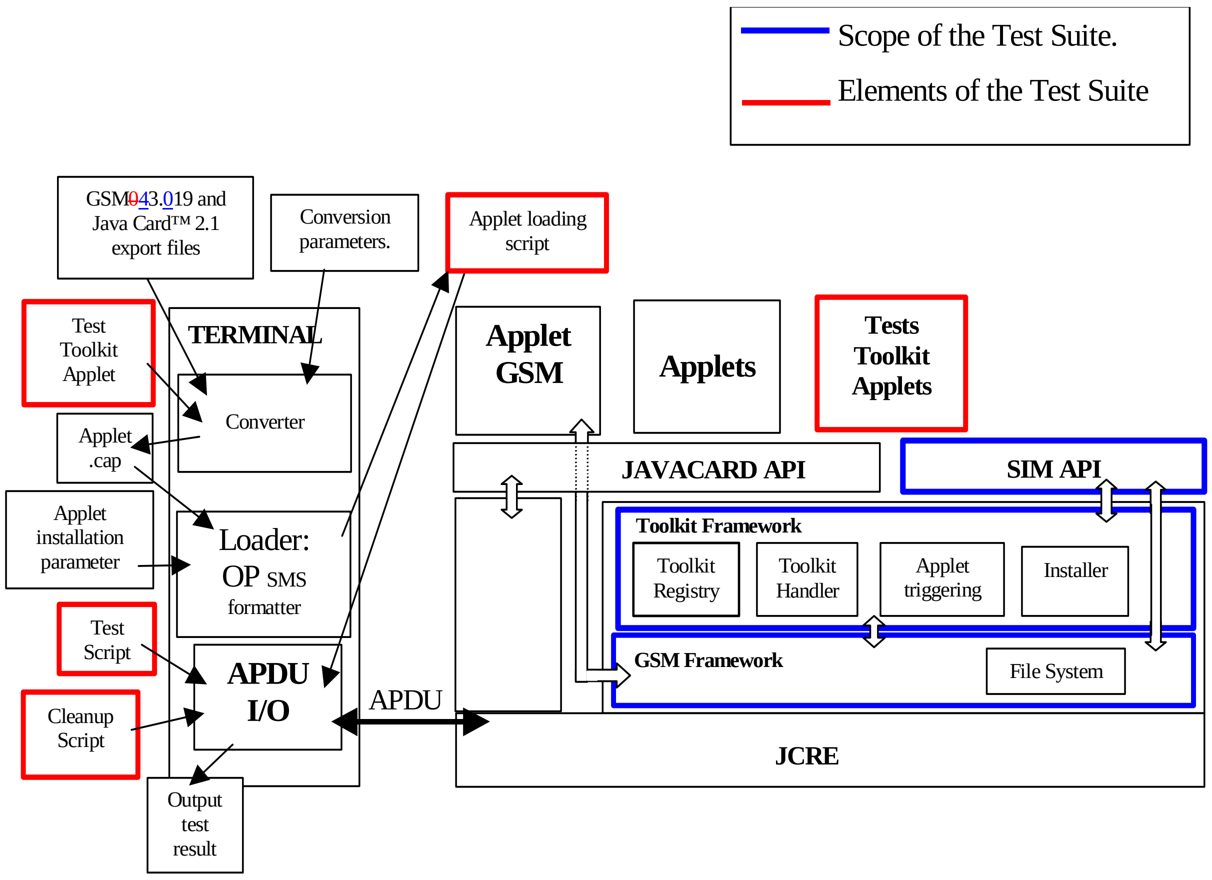
~~This specification contains tests that test interoperability at the API level. This specification does not currently contain tests for interoperability at the SIM API framework and at the byte code level. These are for further study.~~

The tests defined in this specification are applicable to cards implementing ~~TS 03.19 [7] version 7.4.0~~TS 43.019 [7] unless otherwise stated.

The tests defined in this specification require that the card support the concatenation process with 2 concatenated SMS. Therefore the envelope handler shall support 280 bytes of data.

4.2 Test environment description

The general architecture for the test environment is:

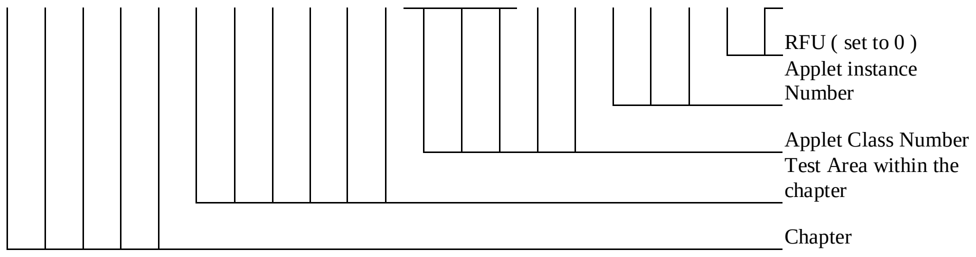


Note: This diagram shows the test architecture required to test interoperability at both API and bytcode level. The latter is currently not included in the current specification. The diagram is for information.

4.6.2 Specific Test Applet Name for Framework

Specific applet test name (bits b4-b24):

b4	b5	b6	b7	b8	b9	b10	b11	b12	b13	b14	b15	b16	b17	b18	b19	b20	b21	b22	b23	b24



for Chapter (5 bits)

00001	Toolkit Installation Parameters
00010	Minimum Handler Availability
00011	Handler Integrity
00100	Applet Triggering
00101	Proactive Command Sending
00110	Framework Security
00111	Envelope Response Posting
01000	File System Context
01001	Exception Handling
01010	Other parts transferred to framework from API
01011	Concatenation processing

other are RFU

Test Area within the chapter (6 bits): values are defined in Annex F

Applet Class number (5 bits): linked to Test Area, it shall start with 1 for classes and shall be 0 for package.

Applet Instance number (3 bits) defined in the test procedure it shall start with 01 for applet instance and shall be 00 for package and class.

6 API Test Plan

6.2 Package sim.toolkit

6.2.2 Interface ToolkitInterface

6.2.2.1.1 Conformance Requirement:

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

```
public void processToolkit(byte event)
        throws  ToolkitException
```

6.2.2.1.1.1 Normal execution

CRRN1: This interface must be implemented by a Toolkit applet (which extends the javacard.framework.Applet class) so that it can be triggered by the Toolkit Handler according to the registration information.

CRRN2: The Toolkit applet will have to implement the processToolkit shared method so that the following events can be notified:

Event	Description
EVENT_PROFILE_DOWNLOAD	Terminal Profile command reception
EVENT_FORMATTED_SMS_PP_ENV	Formatted envelope SMS-PP Data Download reception
EVENT_FORMATTED_SMS_PP_UPD	Formatted Update Record EF SMS
EVENT_FORMATTED_SMS_CB	Formatted envelope Cell Broadcast Data Download command reception
EVENT_UNFORMATTED_SMS_PP_ENV	Unformatted Envelope SMS-PP Data Download reception
EVENT_UNFORMATTED_SMS_PP_UPD	Unformatted Update Record EF SMS
EVENT_UNFORMATTED_SMS_CB	Unformatted Cell Broadcast Data Download command reception
EVENT_MENU_SELECTION	Envelope Menu Selection command reception
EVENT_MENU_SELECTION_HELP_REQUEST	Envelope Menu Selection Help Request command reception
EVENT_CALL_CONTROL_BY_SIM	Envelope Call Control by SIM command reception
EVENT_MO_SHORT_MESSAGE_CONTROL_BY_SIM	Envelope MO Short Message Control by SIM command reception
EVENT_TIMER_EXPIRATION	Envelope Timer Expiration
EVENT_EVENT_DOWNLOAD_MT_CALL	Envelope Event Download - MT call
EVENT_EVENT_DOWNLOAD_CALL_CONNECTED	Envelope Event Download - Call connected
EVENT_EVENT_DOWNLOAD_CALL_DISCONNECTED	Event Download - Call disconnected
EVENT_EVENT_DOWNLOAD_LOCATION_STATUS	Envelope Event Download - Location status
EVENT_EVENT_DOWNLOAD_USER_ACTIVITY	Envelope Event Download - User activity
EVENT_EVENT_DOWNLOAD_IDLE_SCREEN_AVAILABLE	Envelope Event Download - Idle screen available
EVENT_EVENT_DOWNLOAD_CARD_READER_STATUS	Envelope Event Download - Card Reader Status
EVENT_EVENT_DOWNLOAD_LANGUAGE_SELECTION	Envelope Event Download – Language Selection
EVENT_EVENT_DOWNLOAD_BROWSER_TERMINATION	Envelope Event Download – Browser Termination
EVENT_EVENT_DOWNLOAD_DATA_AVAILABLE	Envelope Event Download - Data Available
EVENT_EVENT_DOWNLOAD_CHANNEL_STATUS	Envelope Event Download - Channel Status
EVENT_FIRST_COMMAND_AFTER_SELECT	First command performed after select GSM application or ATR
EVENT_STATUS_COMMAND	Status APDU command event
EVENT_UNRECOGNIZED_ENVELOPE	Unrecognized Envelope command reception

6.2.4 Class EnvelopeHandler

6.2.4.3 Method getSecuredDataLength

Test Area Reference: API_2_ENH_GSDL

6.2.4.3.1 Conformance Requirement:

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

```
public short getSecuredDataLength()
                           throws ToolkitException
```

6.2.4.3.1.1 Normal execution

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

CRRN2: The length is from the first SMS TPDU TLV or Cell Broadcast Page Simple 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 ~~GSM03.48.~~TS 23.048 [8].

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 ~~GSM03.48.TS~~ 23.048 [8].

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 23.048 [8].

CRRN7: If the method is successful and if the event is EVENT_FORMATTED_SMS_PP_ENV, the selected TLV should be the SMS TPDU TLV.

CRRN8: If the method is successful and if the event is EVENT_FORMATTED_SMS_PP_UPD, the selected TLV should be the SMS TPDU TLV.

CRRN9: If the method is successful and if the event is EVENT_FORMATTED_SMS_CB, the selected TLV should be the Cell Broadcast Page TLV.

~~Parameters error~~

~~No requirements~~

6.2.4.3.1.2 Context errors

CRRC1: The method shall thrown ToolkitException (UNAVAILABLE_ELEMENT) in case of unavailable SMS TPDU TLV ~~element.~~element or Cell Broadcast Page Simple TLV

CRRC2: The method shall thrown ToolkitException (UNAVAILABLE_ELEMENT) in case of ~~missing Secured Data.~~wrong data format.

6.2.4.3.2 Test suite files

Specific triggering:

- FORMATTED SMS CB
- UNFORMATTED SMS CB
- FORMATTED SMS PP UPD
- UNFORMATED SMS PP ENV

- For Formatted triggering if CC/RC/DS is used, the security parameters are the one used for downloading applications.

Test Script:	API_2_ENH_GSDL_1.scr
Test Applet:	API_2_ENH_GSDL_1.java
Load Script:	API_2_ENH_GSDL_1.ldr
Cleanup Script:	~~API_2_ENH_GSLD_1.clr~~
~~Parameter File:~~	~~API_2_ENH_GSLD_1.par~~API_2_ENH_GSDL_1.clr
Parameter File:	API_2_ENH_GSDL_1.par

6.2.4.3.3 Test procedure

Id	Description	API Expectation	APDU Expectation
	FORMATTED SMS PP ENV Triggering		

~~1~~	~~Test with FORMATTED_SMS_PP_ENV and TP-OA length of 2~~	~~Returns 0x2A~~	
1	Test with FORMATTED_SMS_PP_ENV and TP-OA length of 2	Returns 0x002A	
~~2~~	~~Test with TP-OA length of 6~~	~~Returns 0x2A~~	
2	Test with TP-OA length of 6	Returns 0x002A	
~~3~~	~~Test with TP-OA length of 12~~	~~Returns 0x2A~~	
3	Test with TP-OA length of 12	Returns 0x002A	
~~4~~	~~Test with RC/CC/DS length of 0~~	~~Returns 0x10~~	
4	Test with RC/CC/DS length of 0	Returns 0x0010	
~~5~~	~~Test with RC/CC/DS length of 8~~	~~Returns 0x10~~	
5	Test with RC/CC/DS length of 8	Returns 0x0010	
~~6~~	~~Test with PCNTR = 0~~	~~Returns 0x10~~	
6	Test with PCNTR = 0	Returns 0x0010	
~~7~~	~~Test with PCNTR = 7~~	~~Returns 0x05~~	
7	Test with PCNTR = 7	Returns 0x0005	
~~8~~	~~Test with SecuredDataLength = 00~~	~~Returns 0x00~~	
8	Test with Secured Data Length = 00	Returns 0x0000	
~~9~~	~~Test with UserDataLength = 0x33~~	~~Returns 0x33~~	
9	Test with Secured Data Length = 0x33	Returns 0x0033	
~~10~~	~~Test with UserDataLength = 0x 6C~~	~~Returns 0x 6C~~	
10	Test with Secured Data Length = 0x6C (UDL = 0x7F)	Returns 0x006C	
~~11~~	~~Test with UserDataLength = 0x 6D~~	~~Returns 0x 6D~~	
11	Test with Secured Data Length = 0x6D (UDL = 0x80)	Returns 0x006D	
~~12~~	~~Test with UserDataLength = maximum length: 0x79~~	~~Returns 0x 79~~	
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 0x05~~	
~~14~~	~~Same test as 1 but with FORMATTED_SMS_PP_UPD~~	~~Returns 0x2A~~	
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		
~~15~~	~~Same test as 2 but with FORMATTED_SMS_PP_UPD~~	~~Returns 0x2A~~	
18	Same test as 1 but with FORMATTED_SMS_PP_UPD	Returns 0x002A	
~~16~~	~~Same test as 3 but with FORMATTED_SMS_PP_UPD~~	~~Returns 0x2A~~	
19	Same test as 2 but with FORMATTED_SMS_PP_UPD	Returns 0x002A	
~~17~~	~~Same test as 4 but with FORMATTED_SMS_PP_UPD~~	~~Returns 0x10~~	
20	Same test as 3 but with FORMATTED_SMS_PP_UPD	Returns 0x002A	
~~18~~	~~Same test as 5 but with FORMATTED_SMS_PP_UPD~~	~~Returns 0x10~~	
21	Same test as 4 but with FORMATTED_SMS_PP_UPD	Returns 0x0010	

~~19~~	~~Same test as 6 but with FORMATTED_SMS_PP_UPD~~	~~Returns 0x10~~	
22	Same test as 5 but with FORMATTED_SMS_PP_UPD	Returns 0x0010	
~~20~~	~~Same test as 7 but with FORMATTED_SMS_PP_UPD~~	~~Returns 0x05~~	
23	Same test as 6 but with FORMATTED_SMS_PP_UPD	Returns 0x0010	
~~21~~	~~Same test as 8 but with FORMATTED_SMS_PP_UPD~~	~~Returns 0x00~~	
24	Same test as 7 but with FORMATTED_SMS_PP_UPD	Returns 0x0005	
~~22~~	~~Same test as 9 but with FORMATTED_SMS_PP_UPD~~	~~Returns 0x33~~	
25	Same test as 8 but with FORMATTED_SMS_PP_UPD	Returns 0x0000	
~~23~~	~~Same test as 10 but with FORMATTED_SMS_PP_UPD~~	~~Returns 0x 6C~~	
26	Same test as 9 but with FORMATTED_SMS_PP_UPD	Returns 0x0033	
~~24~~	~~Same test as 11 but with FORMATTED_SMS_PP_UPD~~	~~Returns 0x 6D~~	
27	Same test as 10 but with FORMATTED_SMS_PP_UPD	Returns 0x006C	
~~25~~	~~Same test as 12 but with FORMATTED_SMS_PP_UPD~~	~~Returns 0x 79~~	
28	Same test as 11 but with FORMATTED_SMS_PP_UPD	Returns 0x006D	
~~26~~	~~Same test as 13 but with FORMATTED_SMS_PP_UPD~~	~~Returns 0x05~~	
29	Same test as 12 but with FORMATTED_SMS_PP_UPD	Returns 0x0079	
~~27~~	~~Same test as 4 but with FORMATTED_SMS_CB~~	~~Returns 0x10~~	
30	Same test as 13 but with FORMATTED_SMS_PP_UPD	Returns 0x0005	
31	Test with secured data length = 0x7F (2 concatenated envelopes are needed)	Returns 0x007F	
32	Test with secured data length = 0x80 (2 concatenated envelopes are needed)	Returns 0x0080	
33	Test with secured data length = maximum length for 2 concatenated envelopes : 0xFA	Returns 0x00FA	
34	Test with FORMATTED_SMS_PP_UPD Verify after call of the method the current TLV is the TPDU TLV: findTLV device identities, getSecuredDataLength and then getValueByte to verify that the current TLV is the TPDU TLV	getValueByte returns 0x0040	
	FORMATTED SMS CB Triggering		
~~28~~	~~Same test as 5 but with FORMATTED_SMS_CB~~	~~Returns 0x10~~	
35	Same test as 4 but with FORMATTED_SMS_CB	Returns 0x0010	
~~29~~	~~Same test as 6 but with FORMATTED_SMS_CB~~	~~Returns 0x10~~	
36	Same test as 5 but with FORMATTED_SMS_CB	Returns 0x0010	
~~30~~	~~Same test as 7 but with FORMATTED_SMS_CB~~	~~Returns 0x05~~	
37	Same test as 6 but with FORMATTED_SMS_CB	Returns 0x0010	
~~31~~	~~Same test as 8 but with FORMATTED_SMS_CB~~	~~Returns 0x00~~	
38	Same test as 7 but with FORMATTED_SMS_CB	Returns 0x0005	
~~32~~	~~Same test as 9 but with FORMATTED_SMS_CB~~	~~Returns 0x33~~	
39	Same test as 8 but with FORMATTED_SMS_CB	Returns 0x0000	
~~33~~	~~Same test as 12 but with maximum length: 0x42, and FORMATTED_SMS_CB~~	~~Returns 0x 42~~	
40	Same test as 9 but with FORMATTED_SMS_CB	Returns 0x0033	
~~34~~	~~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 0x40~~	

~~35~~	~~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 0x58~~	
41	Same test as 12 but with maximum secured data length: 0x42, and FORMATTED_SMS_CB	Returns 0x0042	
~~36~~	~~Send an envelope SMS CB, getSecuredDataLength~~	~~ToolkitException UNAVAILABLE_ELEMENT~~	
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	
	Error tests		
43	Send an envelope SMS CB, getSecuredDataLength	ToolkitException UNAVAILABLE_ELEMENT	
~~37~~	~~Send an envelope SMS PP unformatted~~	~~ToolkitException UNAVAILABLE_ELEMENT~~	
44	Send an envelope SMS PP unformatted	ToolkitException UNAVAILABLE_ELEMENT	

6.2.4.3.4 Test Coverage

~~This method has only been tested with call control and the tests shall be improved during 03.48 tests.~~

CRR number	Test case number
~~N1~~	~~1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13~~
~~N2~~	~~13~~
~~N3~~	~~6, 7~~
~~N4~~	~~1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 25~~
~~N5~~	~~14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26~~
~~N6~~	~~27, 28, 29, 30, 31, 32, 33~~
~~N7~~	~~34~~
~~N8~~	~~35~~
~~C1~~	~~36~~
~~C2~~	~~37~~
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	17
N8	34
N9	42
C1	43
C2	44

6.2.4.4 Method getSecuredDataOffset

Test Area Reference: API_2_ENH_GSDO

6.2.4.4.1 Conformance Requirement:

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

```
public short getSecuredDataOffset()
                           throws ToolkitException
```

6.2.4.4.1.1 Normal execution

CRRN1: The method shall return the offset of the secured data first byte contained 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 and if the SMS TP-UD is formatted according to ~~GSM03.48.~~TS 23.048 [8].

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 ~~GSM03.48.~~TS 23.048 [8].

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 23.048 [8].

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.

~~Parameters error~~

~~No requirements~~CRNN9: If the Secured Data length is zero the value returned shall be the offset of the first byte following the TS 23.048 [8] Command Packet structure.

6.2.4.4.1.2 Context errors

CRRC1: The method shall thrown ToolkitException (UNAVAILABLE_ELEMENT) in case of unavailable SMS TPDU TLV element.

CRRC2: The method shall thrown ToolkitException (UNAVAILABLE_ELEMENT) in case of ~~missing Secured Data.~~wrong data format

6.2.4.4.2 Test suite files

Specific triggering:

- FORMATTED SMS CB
- UNFORMATTED SMS CB
- FORMATTED SMS PP UPD
- UNFORMATED SMS PP ENV

```
-   For Formatted triggering if CC/RC/DS is used, the security parameters are the one used for
downloading applications.
```

Test Script:	API_2_ENH_GSDO_1.scr
Test Applet:	API_2_ENH_GSDO_1.java
Load Script:	API_2_ENH_GSDO_1.ldr
Cleanup Script:	API_2_ENH_GSDO_1.clr
Parameter File:	API_2_ENH_GSDO_1.par

6.2.4.4.3 Test Procedure

Id	Description	API Expectation	APDU Expectation
	FORMATTED SMS PP ENV triggering		
~~1~~	~~Test with TP-OA length of 2~~	~~Returns 0x21~~	
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~~	~~Returns 0x23~~	
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~~	~~Returns 0x26~~	
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 0~~	~~Returns 0x21~~	
4	Test with RC/CC/DS length of 0 and TP-OA length is 2	Returns 0x21	
~~5~~	~~Test with RC/CC/DS length of 8~~	~~Returns 0x29~~	
5	Test with RC/CC/DS length of 8 and TP-OA length is 2	Returns 0x29	
6	Send a SMS PP with 2 TPDU TLV and inside two different secured data offsets	Returns 0x24 (the first offset)	
~~7~~	~~Same test as 1 but with FORMATTED_SMS_PP_UPD~~	~~Returns 0x21~~	
7	Same test as 1 but without any secured data	Returns 0x21	
~~8~~	~~Same test as 2 but with FORMATTED_SMS_PP_UPD~~	~~Returns 0x23~~	
8	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	
~~9~~	~~Same test as 3 but with FORMATTED_SMS_PP_UPD~~	~~Returns 0x26~~	
9	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		
~~10~~	~~Same test as 4 but with FORMATTED_SMS_PP_UPD~~	~~Returns 0x21~~	
10	Same test as 1 but with FORMATTED_SMS_PP_UPD	Returns 0x21	
~~11~~	~~Same test as 5 but with FORMATTED_SMS_PP_UPD~~	~~Returns 0x29~~	
11	Same test as 2 but with FORMATTED_SMS_PP_UPD	Returns 0x23	
~~12~~	~~Same test as 6 but with FORMATTED_SMS_PP_UPD~~	~~Returns 0x24 (the first offset)~~	
12	Same test as 3 but with FORMATTED_SMS_PP_UPD	Returns 0x26	
13	Same test as 4 but with FORMATTED_SMS_PP_UPD	Returns 0x21	
14	Same test as 5 but with FORMATTED_SMS_PP_UPD	Returns 0x29	
15	Same test as 6 but with FORMATTED_SMS_PP_UPD	Returns 0x24 (the first offset)	
~~13~~	~~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~~	
16	Same test as 7 but with FORMATTED_SMS_PP_UPD	Returns 0x21	
17	Test with FORMATTED_SMS_PP UPD Verify after call of the method the current TLV is	Returns 0x40	

	the TPDU TLV: findTLV device identities, getSecuredDataOffset and then getValueByte to verify that the current TLV is the TPDU TLV		
~~14~~	~~Same test as 4 but with FORMATTED_SMS_CB~~	~~Returns 0x16~~	
18	Same test as 10, but with a concatenated SMS (2 Short Messages and maximum Secured Data Length = 0x00FA)	Returns 0x21	
	FORMATTED SMS CB triggering		
19	Same test as 4 but with FORMATTED_SMS_CB	Returns 0x16	
~~15~~	~~Same test as 5 but with FORMATTED_SMS_CB~~	~~Returns 0x1E~~	
20	Same test as 5 but with FORMATTED_SMS_CB	Returns 0x1E	
~~16~~	~~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 0x58~~	
21	Same test as 7 but with FORMATTED_SMS_CB	Returns 0x16	
~~17~~	~~Send an UNFORMATTED SMS CB envelope, getSecuredDataOffset~~	~~ToolkitException UNAVAILABLE_ELEMENT~~	
22	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 Triggering		
23	Send an UNFORMATTED SMS CB envelope, getSecuredDataOffset	ToolkitException UNAVAILABLE_ELEMENT	
~~18~~	~~Send an UNFORMATTED SMS PP envelope~~	~~ToolkitException UNAVAILABLE_ELEMENT~~	
24	Send an UNFORMATTED SMS PP envelope, getSecuredDataOffset	ToolkitException UNAVAILABLE_ELEMENT	
~~19~~	~~Send an FORMATTED SMS PP envelope with no secured data , getSecuredDataOffset~~	~~Returns 0x21~~	

6.2.4.4.4 Test Coverage

~~This method has only been tested with call control and the tests shall be improved during 03.48 tests.~~

CRR number	Test case number
~~N1~~	~~1, 2, 3, 4, 5, 6, 19~~
~~N2~~	~~6, 12~~
~~N3~~	~~1, 2, 3, 4, 5, 6, 19~~
~~N4~~	~~7, 8, 9, 10, 11~~
~~N5~~	~~14,15~~
~~N6~~	~~13~~
~~N7~~	~~16~~
~~C1~~	~~17~~
~~C2~~	~~18~~
N1	1 to 22.
N2	6, 15.
N3	1 to 9.
N4	10 to 18.
N5	19, 20, 21, 22
N6	8
N7	17
N8	22
N9	7, 16, 21.
C1	23
C2	24

6.2.4.5 Method getTheHandler

Test Area Reference: API_2_ENH_GTHD

6.2.4.5.1 Conformance Requirements

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

```
public static EnvelopeHandler getTheHandler()
                                  throws ToolkitException
```

6.2.4.5.1.1 Normal execution

CRRN1: The method shall return the single system instance of the EnvelopeHandler class.

CRRN2: The EnvelopeHandler is a Temporary JCRE Entry Point Object (~~-~~see Javacard 2.1 Runtime Environment (JCRE) Specification [12])

~~Parameters error~~

~~No requirements~~

6.2.4.5.1.2 Context errors

CRRC1: The method shall thrown ToolkitException (HANDLER_NOT_AVAILABLE) if the handler is busy.

6.2.4.5.2 Test suite files

Test Script:	API_2_ENH_GTHD_1.scr
Test Applet:	API_2_ENH_GTHD_1.java
Load Script:	API_2_ENH_GTHD_1.ldr
Cleanup Script:	API_2_ENH_GTHD_1.clr
Parameter File:	API_2_ENH_GTHD_1.par

6.2.4.5.3 Test Procedure

Id	Description	API Expectation	APDU Expectation
1	getTheHandler twice	The returned objects shall be the same	
~~2~~	~~Verify that getTheHandler returns an EnvelopeHandler~~ ~~GetTheHandler~~	~~The reference returned shall be an EnvelopeHandler (checkcast)~~	
2	Verify that getTheHandler returns an EnvelopeHandler GetTheHandler	The reference returned shall be an EnvelopeHandler (check cast)	
3	Verify the returned value is not null GetTheHandler	The reference returned shall not be null.	

6.2.4.5.4 Test Coverage

CRR number	Test case number
N1	1, 2, 3
N2	To be checked in Framework tests and insert here cross reference
C1	To be checked in Framework tests and insert here cross reference

6.2.4.6 Method getTPUDLOffset

Test Area Reference: API_2_ENH_GTPO

6.2.4.6.1 Conformance Requirement:

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

```
public short getTPUDLOffset()
                     throws ToolkitException
```

6.2.4.6.1.1 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.

~~Parameters error~~

~~No requirements~~

6.2.4.6.1.2 Context errors

CRRC1: The method shall thrown ToolkitException (UNAVAILABLE_ELEMENT) in case of unavailable SMS TPDU TLV element.

CRRC2: The method shall thrown ToolkitException (UNAVAILABLE_ELEMENT) if the TPUDL field does not exist.

6.2.4.6.2 Test suite files

Specific triggering:

~~FORMATTED SMS PP UPD~~
~~UNFORMATTED SMS PP UPD~~

- FORMATTED SMS PP UPD

- UNFORMATTED SMS PP UPD
- UNFORMATTED SMS PP ENV
- UNFORMATTED SMS CB

Test Script:	API_2_ENH_GTPO_1.scr
Test Applet:	API_2_ENH_GTPO_1.java
Load Script:	API_2_ENH_GTPO_1.ldr
Cleanup Script:	API_2_ENH_GTPO_1.clr
Parameter file:	API_2_ENH_GTPO_1.par

6.2.4.6.3 Test procedure

Id	Description	API 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)~~	
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 FORMATTED_SMS_PP_UPD~~	~~Returns 0x0D~~	
5	Same test as 1, but with a concatenated SMS (2 Short Messages and maximum Secured Data Length = 0x00FA)	Returns 0x0D	
~~6~~	~~Same test as 2 but with FORMATTED_SMS_PP_UPD~~	~~Returns 0x0F~~	
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 3 but with FORMATTED_SMS_PP_UPD~~	~~Returns 0x12~~	
7	Same test as 1 but with FORMATTED_SMS_PP_UPD	Returns 0x0D	
~~8~~	~~Same test as 4 but with FORMATTED_SMS_PP_UPD~~	~~Returns 0x10 (the first offset)~~	
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	
~~9~~	~~Same test as 1 but with UNFORMATTED_SMS_PP_UPD~~	~~Returns 0x0D~~	
~~10~~	~~Same test as 2 but with UNFORMATTED_SMS_PP_UPD~~	~~Returns 0x0F~~	
10	Same test as 4 but with FORMATTED_SMS_PP_UPD	Returns 0x10 (the first offset)	
~~11~~	~~Same test as 3 but with UNFORMATTED_SMS_PP_UPD~~	~~Returns 0x12~~	
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 4 but with UNFORMATTED_SMS_PP_UPD~~	~~Returns 0x12 (the first offset)~~	
12	Same test as 1 but with UNFORMATTED_SMS_PP_UPD	Returns 0x0D	

~~13~~	~~Same test as 1 but with UNFORMATTED_SMS_PP_ENV~~	~~Returns 0x0D~~	
13	Same test as 2 but with UNFORMATTED_SMS_PP_UPD	Returns 0x0F	
~~14~~	~~Same test as 2 but with UNFORMATTED_SMS_PP_ENV~~	~~Returns 0x0F~~	
14	Same test as 3 but with UNFORMATTED_SMS_PP_UPD	Returns 0x12	
~~15~~	~~Same test as 3 but with UNFORMATTED_SMS_PP_ENV~~	~~Returns 0x12~~	
15	Same test as 4 but with UNFORMATTED_SMS_PP_UPD	Returns 0x12 (the first offset)	
~~16~~	~~Same test as 4 but with UNFORMATTED_SMS_PP_ENV~~	~~Returns 0x10 (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	
~~17~~	~~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~~	
~~18~~	~~Send an envelope SMS CB, getTPUDLOffset~~	~~ToolkitException UNAVAILABLE_ELEMENT~~	
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	
	SMS CB triggering		
22	Send an envelope SMS CB, getTPUDLOffset	ToolkitException UNAVAILABLE_ELEMENT	

6.2.4.6.4 Test Coverage

CRR number	Test case number
~~N1~~	~~1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17~~
~~N2~~	~~4~~
N1	1 to 21.
N2	4, 10, 15, 20.
~~N3~~	~~1, 2, 3, 4, 17~~
~~N4~~	~~5, 6, 7, 8~~
~~N5~~	~~13, 14, 15, 16~~
N3	1, 2, 3, 4, 5, 6
N4	7, 8, 9, 10, 11,
N5	12, 13, 14, 15, 16
~~N6~~	~~9, 10, 11, 12~~
~~N7~~	~~17~~
~~C1~~	~~18~~
N6	17, 18, 19, 20, 21
N7	6
C1	22
~~C2~~	~~Don't no how to test~~
C2	Not applicable

6.2.4.7 Method getLength

Test Area Reference: API_2_ENH_GLEN

6.2.4.7.1 Conformance Requirement

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

```
public short getLength()
                  throws ToolkitException
```

6.2.4.7.1.1 Normal execution

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

~~Parameter Error~~

~~No requirements~~

6.2.4.7.1.2 Context errors

CRRC1: if the handler is busy an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException.HANDLER_NOT_AVAILABLE.

6.2.4.7.2 Test Suite files

Specific triggering: None

Test Script:	API_2_ENH_GLEN_1.scr
Test Applet:	API_2_ENH_GLEN_1.java
Load Script:	API_2_ENH_GLEN_1.ldr
Cleanup Script:	API_2_ENH_GLEN_1.clr
Parameter File:	API_2_ENH_GLEN_1.par

6.2.4.7.3 Test Procedure

Id	Description	API Expectation	APDU Expectation
1	Send an envelope SMS PP with BER length of ~~-~~ 0x31	Result of getLength() is 0x0031	
2	Send an envelope SMS PP with BER length of ~~-~~ 0x7F	Result of getLength() is 0x007Fh	
3	Send an envelope SMS PP with BER length of ~~-~~ 81 80	Result of getLength() is 0x0080h	
~~4~~	~~Send an envelope SMS PP with BER length of - 81 FC~~	~~Result of getLength() is 0x00FCh~~	
4	Send an envelope SMS PP with BER length of 81 FC (maximum length for a single SMS)	Result of getLength() is 0x00FCh	
5	Send formatted SMS with BER length of 0x00FF, using 2 concatenated SMS	Result of getLength() is 0x00FFh	
6	Send formatted SMS with BER length of 0x0100, using 2 concatenated SMS	Result of getLength() is 0x0100h	
7	Send formatted SMS with maximum user data length (0x10D) (BER length:0x012F), using 2 concatenated SMS	Result of getLength() is 0x012Fh	

6.2.4.7.4 Test Coverage

CRR number	Test case number
~~N1~~	~~1, 2, 3, 4~~
N1	1, 2, 3, 4, 5, 6, 7
C1	Does not apply for EnvelopeHandler

6.2.4.8 Method copy

Test Area Reference: API_2_ENH_COPY_BSS

6.2.4.8.1 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
```

6.2.4.8.1.1 Normal execution

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

CRRN2: returns dstOffset + dstLength.

6.2.4.8.1.2 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 simple TLV List, an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException OUT_OF_TLV_BOUNDARIES.

6.2.4.8.1.3 Context errors

CRRC1: if the handler is busy an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException HANDLER_NOT_AVAILABLE.

6.2.4.8.2 Test Suite files

Specific triggering: None

Test Script:	API_2_ENH_COPY_BSS_1.scr
Test Applet:	API_2_ENH_COPY_BSS_1.java
Load Script:	API_2_ENH_COPY_BSS_1.ldr
Cleanup Script:	~~API_2_ENH_COPY_BSS_1.clr~~
~~Parameter File:~~	~~API_2_ENH_COPY_BSS_1.par~~API_2_ENH_COPY_BSS_1.clr
Parameter File:	API_2_ENH_COPY_BSS_1.par

6.2.4.8.3 Test Procedure

Id	Description	API Expectation	APDU Expectation
1	**NULL as parameter to dstBuffer**	NullPointerException is thrown	

2	**dstOffset ≥ dstBuffer.length**
	`dstBuffer.length = 5`
	`dstOffset = 5`
	`dstLength = 1`
		ArrayIndexOutOfBoundsException is thrown	

3	**dstOffset < 0**
	`dstBuffer.length = 5`
	`dstOffset = -1`
	`dstLength = 1`
		ArrayIndexOutOfBoundsException is thrown	

4	**dstLength > dstBuffer.length**
	`dstBuffer.length = 5`
	`dstOffset = 0`
	`dstLength = 6`
		ArrayIndexOutOfBoundsException is thrown	

5	**DstOffset + dstLength > dstBuffer.length**
	`DstBuffer.length = 5`
	`DstOffset = 3`
	`DstLength = 3`
		ArrayIndexOutOfBoundsException is thrown	

6	**dstLength < 0**
	`dstBuffer.length = 5`
	`dstOffset = 0`
	`dstLength = -1`
		ArrayIndexOutOfBoundsException is thrown	

7	**DstLength > length of the simple TLV list**
	`DstBuffer.length = 48`
	`DstOffset = 0`
	`DstLength = 48`
		ToolkitException.OUT_OF_TLV_BOUNDARIES is thrown	

8	**Successful call, dstBuffer is the whole buffer**
	`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**
	`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**
	`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**
	`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	

	Send a Formatted SMS PP with the maximum user data length = 0x010D, using 2 concatenated envelopes		

17	**Successful call, copy with length =299**
	`dstBuffer.length = 299`
	`dstOffset = 0`
	`dstLength = 299`
		Result of copy() is 0x12B	

6.2.4.8.4 Test Coverage

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

6.2.4.9 Method findTLV

Test Area Reference: API_2_ENH_FINDBB

6.2.4.9.1 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
```

6.2.4.9.1.1 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.

6.2.4.9.1.2 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.

6.2.4.9.1.3 Context errors

CRRC1: if the handler is busy an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException HANDLER_NOT_AVAILABLE.

6.2.4.9.2 Test Suite files

Specific triggering: None

Test Script:	API_2_ENH_~~ENH_~~FINDBB_1.scr
Test Applet:	~~API_2_ENH_ENH_FINDBB_1.java~~
~~Load Script:~~	~~API_2_ENH_ENH_FINDBB_1.ldr~~API_2_ENH_FINDBB_1.java
Load Script:	API_2_ENH_FINDBB_1.ldr
Cleanup Script:	API_2_ENH_FINDBB_1.clr

Parameter File: API_2_ENH_FINDBB_1.par

6.2.4.9.3 Test Procedure

Id	Description	API Expectation	APDU Expectation
	Trig the applet with SMS PP including one more tag 02 and one TAG 04		
1	**Invalid input parameter** `Occurrence = 0`	ToolkitException.BAD_INPUT_PARAMETER is thrown	

2	**Search 1st TLV**
	`Tag = 02h`
	`Occurrence = 1`
		Result is TLV_FOUND_CR_SET	

3	**Call the getValueLength() method**	Result is 0x02	

4	**Search 2nd TLV**
	`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
	`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**
	`Tag = 02h`
	`Occurrence = 3`
		Result is TLV_NOT_FOUND	

9	**Call the getValueLength() method**	ToolkitException.UNAVAILABLE_ELEMENT is thrown.	

10	**Search the TLV**
	`Tag = 02h`
	`Occurrence = 2`
		Result is TLV_FOUND_CR_NOT_SET	

11	**Search the TLV**
	`Tag = 04h`
	`Occurrence = 1`
		Result is TLV_FOUND_CR_NOT_SET	

~~12~~	~~**Search tag 81h**~~ ~~`Tag = 86h`~~ ~~`Occurrence = 1`~~	~~Result is TLV_FOUND_CR_SET~~	

12	**Search tag 86h**
	`Tag = 86h`
	`Occurrence = 1`
		Result is TLV_FOUND_CR_SET	

13	**Search tag 84h**
	`Tag = 84h`
	`Occurrence = 1`
		Result is TLV_FOUND_CR_NOT_SET	

6.2.4.9.4 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 EnvelopeHandler

6.2.4.10 Method getValueLength

Test Area Reference: API_2_ENH_GVLE~~.~~

6.2.4.10.1 Conformance Requirement

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

```
public short getValueLength()
                         throws ToolkitException
```

6.2.4.10.1.1 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.

~~6.2.4.10.1.2 Parameter errors~~

~~No requirements.~~

6.2.4.10.1.2 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.

6.2.4.10.2 Test Suite files

Specific triggering: None

Test Script:	API_2_ENH_GVLE_1.scr
Test Applet:	~~API_2_ENH_GVLE_1.java~~
~~Load Script:~~	~~API_2_ENH_GVLE_1.ldr~~
~~Cleanup Script:~~	~~API_2_ENH_GVLE_1.clr~~
~~Parameter File:~~	API_2_ENH_GVLE_1.java
Load Script:	API_2_ENH_GVLE_1.ldr
Cleanup Script:	API_2_ENH_GVLE_1.clr
Parameter File:	API_2_ENH_GVLE_1.par

6.2.4.10.3 Test Procedure

Id	Description	API Expectation	APDU Expectation
	Fill the SMS PP with TLV: 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	
	Send Formatted SMS PP with the maximum user data length = 0x010D, using 2 concatenated envelopes		
5	**Search SMS TPDU TAG**		
	`getValueLength()`	Result is 0X0120	

6.2.4.10.4 Test Coverage

CRR number	Test case number
~~N1~~	~~2, 3, 4~~
N1	2, 3, 4, 5
C1	Does not apply for EnvelopeHandler
C2	1

6.2.4.11 Method getValueByte

Test Area Reference: API_2_ENH_GVBYS~~.~~

6.2.4.11.1 Conformance Requirement

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

```
public byte getValueByte(short valueOffset)
                  throws ToolkitException
```

6.2.4.11.1.1 Normal execution

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

6.2.4.11.1.2 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.

6.2.4.11.1.3 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.

6.2.4.11.2 Test Suite files

Specific triggering: None

Test Script:	API_2_ENH_GVBYS_1.scr
Test Applet:	~~API_2_ENH_GVBYS_1.java~~
~~Load Script:~~	~~API_2_ENH_GVBYS_l.dr~~
~~Cleanup Script:~~	~~API_2_ENH_GVBYS_1.clr~~
~~Parameter File:~~	API_2_ENH_GVBYS_1.java
Load Script:	API_2_ENH_GVBYS_1.ldr
Cleanup Script:	API_2_ENH_GVBYS_1.clr
Parameter File:	API_2_ENH_GVBYS_1.par

6.2.4.11.3 Test Procedure

Id	Description	API Expectation	APDU Expectation
	Fill the SMS PP with TLV: 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	
	Send Formatted SMS PP with the maximum user data length = 0x010D, using 2 concatenated envelopes		
9	**Search SMS TPDU TAG**		
	`getValueByte(0x011F)`	Result is 0xFA	

6.2.4.11.4 Test Coverage

CRR number	Test case number
~~N1~~	~~3, 4, 5, 6, 7, 8~~
N1	3, 4, 5, 6, 7, 8, 9
P1	2
C1	Does not apply for EnvelopeHandler
C2	1

6.2.4.12 Method copyValue

Test Area Reference: API_2_ENH_CPYVS_BSS~~.~~

6.2.4.12.1 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
```

6.2.4.12.1.1 Normal execution

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

CRRN2: returns~~-~~ dstOffset + dstLength.

6.2.4.12.1.2 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.

6.2.4.12.1.3 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.

6.2.4.12.2 Test Suite files

Specific triggering: None

Test Script:	API_2_ENH_CPYVS_BSS_1.scr
Test Applet:	~~API_2_ENH_CPYVS_BSS_1.java~~
~~Load Script:~~	~~API_2_ENH_CPYVS_BSS_1.ldr~~
~~Cleanup Script:~~	~~API_2_ENH_CPYVS_BSS_1.clr~~
~~Parameter File:~~	API_2_ENH_CPYVS_BSS_1.java
Load Script:	API_2_ENH_CPYVS_BSS_1.ldr
Cleanup Script:	API_2_ENH_CPYVS_BSS_1.clr
Parameter File:	API_2_ENH_CPYVS_BSS_1.par

6.2.4.12.3 Test procedure

Id	Description	API Expectation	APDU Expectation
1	**Search TLV 02h**		
	`copyValue() with a null dstBuffer`	NullPointerException is thrown	
2	**Search TLV 0Bh**		

	dstOffset ≥ dstBuffer.length
	`dstBuffer.length = 5`
	`dstOffset = 5`
	`dstLength = 1`
		ArrayIndexOutOfBoundsException is thrown	

3	**dstOffset < 0**
	`dstBuffer.length = 5`
	`dstOffset = -1`
	`dstLength = 1`
		ArrayIndexOutOfBoundsException is thrown	

4	**dstLength >dstBuffer.length**
	`dstBuffer.length = 5`
	`dstOffset = 0`
	`dstLength = 6`
		ArrayIndexOutOfBoundsException is thrown	

5	**dstOffset + dstLength >dstBuffer.length**
	`dstBuffer.length = 5`
	`dstOffset = 3`
	`dstLength = 3`
		ArrayIndexOutOfBoundsException is thrown	

6	**dstLength < 0**
	`dstBuffer.length = 5`
	`dstOffset = 0`
	`dstLength = -1`
		ArrayIndexOutOfBoundsException is thrown	

7	**Search TLV 06h**		

	valueOffset ≥ TLV Length
	`valueOffset = 6`
	`dstBuffer.length = 15`
	`dstOffset = 0`
	`dstLength = 1`
		ToolkitException.OUT_OF_TLV_BOUNDARIES is thrown	

8	**valueOffset < 0**
	`valueOffset = -1`
	`dstBuffer.length = 15`
		ToolkitException.OUT_OF_TLV_BOUNDARIES is thrown	

	`dstOffset = 0`
	`dstLength = 1`		

9	**dstLength > TLV length**
	`valueOffset = 0`
	`dstBuffer.length = 15`
	`dstOffset = 0`
	`dstLength = 7`
		ToolkitException.OUT_OF_TLV_BOUNDARIES is thrown	

10	**valueOffset + dstLength > TLV length**
	`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	
12	**Search TLV 06h**		

	Successful call
	`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`~~		
14	**initialise dstBuffer** `dstBuffer = 55 55 … 55`		

	Successful call
	`valueOffset = 1`
	`dstBuffer.length = 20`
	`dstOffset = 3`
	`dstLength = 4`
		Result of copyValue() is 0x0007	

15	**Compare buffer**
	`buffer =`
	`55 55 55 11 22`
	`33 44 55 55 55`
	`55 55 55 55 55`
	`55 55 55 55 55`
		Result is 00h	

16	**Successful call, copy with length =0**
	`dstBuffer.length = 20`
	`dstOffset = 20`
	`dstLength = 0`
		Result of copyValue() is 20	

	Send Formatted SMS with the maximum user data length = 0x010D, using 2 concatenated envelopes		
17	**Search SMS TPDU TAG**		

	Successful call
	`valueOffset = 0x11`
	`dstBuffer.length = 0x010D`
	`dstOffset = 0`
	`dstLength = 0x010D`
		Result of copyValue() is 0x010D	

18	**Compare buffer** `buffer = 0348 header and secured data (01 … FA)`	Result is 00h	
19	**Initialise dstBuffer** `dstBuffer = 55 55 … 55`		

	Successful call
	`valueOffset = 0x0111`
	`dstBuffer.length = 0x010D`
	`dstOffset = 0x0100`
	`dstLength = 0x000D`
		Result of copyValue() is 0x010D	

20	**Compare buffer**
	`buffer =`
	`55 55 55 55 55 55 55 55`
	`…`
	`55 55 EE EF F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA`
		Result is 00h	

6.2.4.12.4 Test Coverage

CRR number	Test case number
~~N1~~	~~13, 15~~
N1	13, 15, 18, 20
~~N2~~	~~12, 14, 16~~
N2	12, 14, 16, 17, 19
P1	1
P2	2, 3, 4, 5, 6
P3	7, 8, 9, 10
C1	Does not apply for EnvelopeHandler
C2	11

6.2.4.13 Method compareValue

Test Area Reference: API_2_ENH_CPRVS_BSS~~.~~

6.2.4.13.1 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
```

6.2.4.13.1.1 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 simple TLV List is less than that in compareBuffer.

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

6.2.4.13.1.2 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.

6.2.4.13.1.3 Context errors

CRRC1: if the handler is busy an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException HANDLER_NOT_AVAILABLE.

CRRC2: in case of unavailable TLV element an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException UNAVAILABLE_ELEMENT.

6.2.4.13.2 Test Suite files

Specific triggering: None

Test Script: API_2_ENH_CPRVS_BSS_1.scr

Test Applet: ~~API_2_ENH_CPRVS_BSS_1.java~~

~~Load Script: API_2_ENH_CPRVS_BSS_1.ldr~~

~~Cleanup Script: API_2_ENH_CPRVS_BSS_1.clr~~

~~Parameter File:~~ API_2_ENH_CPRVS_BSS_1.java

Load Script: API_2_ENH_CPRVS_BSS_1.ldr

Cleanup Script: API_2_ENH_CPRVS_BSS_1.clr

Parameter File: API_2_ENH_CPRVS_BSS_1.par

6.2.4.13.3 Test procedure

Id	Description	API Expectation	APDU Expectation
1	**Search TLV 02h**		
	`compareValue() with a null compareBuffer`	NullPointerException is thrown	
2	**Search TLV 0Bh**		

	compareOffset ≥ compareBuffer.length
	`compareBuffer.length = 5`
	`compareOffset = 5`
	`compareLength = 1`
		ArrayIndexOutOfBoundsException is thrown	

3	**compareOffset < 0**
	`compareBuffer.length = 5`
	`compareOffset = -1`
	`compareLength = 1`
		ArrayIndexOutOfBoundsException is thrown	

4	**compareLength >compareBuffer.length**
	`compareBuffer.length = 5`
	`compareOffset = 0`
	`compareLength = 6`
		ArrayIndexOutOfBoundsException is thrown	

5	**compareOffset + compareLength >compareBuffer.length**
	`compareBuffer.length = 5`
	`compareOffset = 3`
	`compareLength = 3`
		ArrayIndexOutOfBoundsException is thrown	

6	**compareLength < 0**
	`compareBuffer.length = 5`
	`compareOffset = 0`
	`compareLength = -1`
		ArrayIndexOutOfBoundsException is thrown	

7	**Search TLV 06h**		

	valueOffset ≥ TLV Length
	`valueOffset = 6`
	`compareBuffer.length = 15`
	`compareOffset = 0`
	`compareLength = 1`
		ToolkitException.OUT_OF_TLV_BOUNDARIES is thrown	

8	**valueOffset < 0**
	`valueOffset = -1`
	`compareBuffer.length = 15`
	`compareOffset = 0`
	`compareLength = 1`
		ToolkitException.OUT_OF_TLV_BOUNDARIES is thrown	

9	**compareLength > TLV length**
	`valueOffset = 0`
	`compareBuffer.length = 15`
	`compareOffset = 0`
	`compareLength = 7`
		ToolkitException.OUT_OF_TLV_BOUNDARIES is thrown	

10	**valueOffset + compareLength > TLV length**
	`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`~~		
	Initialise compareBuffer `compareBuffer =`		

	`81 11 22 33 44 F5`		

	Compare buffers
	`valueOffset = 0`
	`compareOffset = 0`
	`compareLength = 6`
		Result is 00h	

~~13~~	~~**Initialize compareBuffer**~~ ~~`compareBuffer =`~~ ~~`7F 11 22 33 44 F5`~~		

13	**Initialise 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`~~		

14	**Initialise 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`~~		

15	**Initialise compareBuffer**
	`compareBuffer =`
	`55 55 55 81 11 22 33 44 F5`
	`55 55 55 55 55`		

	Compare buffers
	`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`~~		

16	**Initialise 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`~~		

17	**Initialise 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	

	Send Formatted SMS PP with the maximum user data length = 0x010D, using 2 concatenated envelopes		
	Search SMS TPDU TAG		
	Initialise compareBuffer `compareBuffer = 0348 header and formatted data(01 02 … FA)`		

19	**Compare buffers**
	`valueOffset = 0x11`
	`compareOffset = 0`
	`compareLength = 0x010D`
	`compareBufferLength = 0x010D`
		Result is 00h	

20	**Compare buffers**
	`valueOffset = 0x0111`
	`compareOffset = 0x0100`
	`compareLength = 0x000D`
	`compareBufferLength = 0x010D`
		Result is 00h	

6.2.4.13.4 Test Coverage

CRR number	Test case number
~~N1~~	~~12, 15~~
N1	12, 15, 19, 20
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 EnvelopeHandler
C2	11

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

Test Area Reference: API_2_ENH_FACYB_BS~~.~~

6.2.4.14.1 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
```

6.2.4.14.1.1 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.

6.2.4.14.1.2 Parameter errors

CRRP1: if dstBuffer is null NullPointerException shall be thrown.

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

6.2.4.14.1.3 Context errors

CRRC1: if the handler is busy an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException HANDLER_NOT_AVAILABLE.

6.2.4.14.2 Test Suite files

Specific triggering: None

Test Script: API_2_ENH_FACYB_BS_1.scr

Test Applet: ~~API_2_ENH_FACYB_BS_1.java~~

~~Load Script: API_2_ENH_FACYB_BS_1.ldr~~

~~Cleanup Script: API_2_ENH_FACYB_BS_1.clr~~

~~Parameter File:~~ API_2_ENH_FACYB_BS_1.java

Load Script: API_2_ENH_FACYB_BS_1.ldr

Cleanup Script: API_2_ENH_FACYB_BS_1.clr

Parameter File: API_2_ENH_FACYB_BS_1.par

6.2.4.14.3 Test procedure

Id	Description	API Expectation	APDU Expectation
	Fill the SMS PP with TLV: 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**
	`tag = 06h`
	`dstBuffer.length = 06`
	`dstOffset = 06`
		ArrayIndexOutOfBoundsException is thrown	

3	**dstOffset < 0**
	`dstBuffer.length = 06`
	`dstOffset = -1`
		ArrayIndexOutOfBoundsException is thrown	

4	**length > dstBuffer.length**
	`dstBuffer.length = 05`
	`dstOffset = 0`
		ArrayIndexOutOfBoundsException is thrown	

5	**DstOffset + length >dstBuffer.length**
	`DstBuffer.length = 06`
	`DstOffset = 1`
		ArrayIndexOutOfBoundsException is thrown	

6	**Select a TLV (tag 02h)**		

	findAndCopyValue()
	`tag = 03h`
		ToolkitException.UNAVAILABLE_ELEMENT is thrown	

	~~**Call the getValueLength() method**~~	~~ToolkitException.UNAVAILABLE_ELEMENT is thrown.~~	
7	**Call the getValueLength() method**	ToolkitException.UNAVAILABLE_ELEMENT is thrown.	
~~7~~	~~**Successful call**~~ ~~`Tag = 06h`~~ ~~`DstBuffer.length = 06`~~ ~~`DstOffset = 0`~~	~~Result of findAndCopyValue () is 0006~~	

8	**Successful call**
	`Tag = 06h`
	`DstBuffer.length = 06`
	`DstOffset = 0`
		Result of findAndCopyValue () is 0006	

~~8~~	~~**Compare buffer**~~ ~~`buffer = 81 11 22 33 44 F5`~~	~~Result is 00h~~	
9	**Compare buffer** `buffer = 81 11 22 33 44 F5`	Result is 00h	
~~9~~	~~**initialize dstBuffer**~~ ~~`dstBuffer = 55 55 … 55`~~		
10	**initialise dstBuffer** `dstBuffer = 55 55 … 55`		

	Successful call
	`dstBuffer.length = 12`
	`dstOffset = 2`
		Result of findAndCopyValue () is 0008	

~~10~~	~~**Compare buffer**~~ ~~`buffer =`~~ ~~`55 55 81 11 22 33 44 F5 55 55 55 55`~~	~~Result is 00h~~	

11	**Compare buffer**
	`buffer =`
	`55 55 81 11 22 33 44 F5 55 55 55 55`
		Result is 00h	

~~11~~	~~**Successful call**~~ ~~`tag = 02h`~~ ~~`dstBuffer.length = 2`~~ ~~`dstOffset = 0`~~	~~Result of findAndCopyValue () is 0002~~	

12	**Successful call**
	`tag = 02h`
	`dstBuffer.length = 2`
	`dstOffset = 0`
		Result of findAndCopyValue () is 0002	

~~12~~	~~**Compare buffer**~~	~~Result is 00h~~	

	~~buffer = 83 81~~		
13	**Compare buffer** `buffer = 83 81`	Result is 00h	
~~13~~	~~**Successful call (with tag 82h)**~~ ~~`tag = 82h`~~ ~~`dstBuffer.length = 02`~~ ~~`dstOffset = 0`~~	~~Result of findAndCopyValue () is 0002~~	

14	**Successful call (with tag 82h)**
	`tag = 82h`
	`dstBuffer.length = 02`
	`dstOffset = 0`
		Result of findAndCopyValue () is 0002	

~~14~~	~~**Compare buffer**~~ ~~`buffer = 83 81`~~	~~Result is 00h~~	
15	**Compare buffer** `buffer = 83 81`	Result is 00h	
~~15~~	~~**Successful call (with tag B3h)**~~ ~~`tag = B3h`~~ ~~`dstBuffer.length = C4`~~ ~~`dstOffset = 0`~~	~~Result of findAndCopyValue () is 00C4~~	

16	**Successful call (with tag B3h)**
	`tag = B3h`
	`dstBuffer.length = C4`
	`dstOffset = 0`
		Result of findAndCopyValue () is 00C4	

~~16~~	~~**Compare buffer**~~ ~~`buffer = 01 02 … C4`~~	~~Result is 00h~~	
17	**Compare buffer** `buffer = 01 02 … C4`	Result is 00h	
	Send Formatted SMS PP with the maximum user data length = 0x010D, using 2 concatenated envelopes		

18	**Successful call (with SMS TPDU TAG)**
	`tag = 0Bh`
	`dstBuffer.length = 0x011E`
	`dstOffset = 0`
		Result of findAndCopyValue () is 0x011E	

19	**Compare buffer** `buffer = 0348 Header + secured data (01 02 … FA)`	Result is 00h	

20	**Successful call (with SMS TPDU TAG)**
	`tag = 0Bh`
	`dstBuffer.length = 0x0220`
	`dstOffset = 0x0100`
		Result of findAndCopyValue () is 0x021E	

21	**Compare buffer** `buffer = 0348 Header + secured data (01 02 … FA)`	Result is 00h	

6.2.4.14.4 Test Coverage

CRR number	Test case number
~~N1~~	~~8, 10, 12~~
~~N2~~	~~6~~
~~N3~~	~~7, 9, 11~~
~~N4~~	~~13, 14, 15, 16~~
N1	9, 11, 13
N2	6, 7
N3	8, 10, 12
N4	14, 15, 16, 17, 18, 19, 20, 21
P1	1
P2	2, 3, 4, 5
C1	Does not apply for EnvelopeHandler

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

Test Area Reference: API_2_ENH_FACYBS_BSS~~.~~

6.2.4.15.1 Conformance requirement

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

```
public short findAndCopyValue(byte tag,
                              byte occurence,
                              short valueOffset,
                              byte[] dstBuffer,
                              short dstOffset,
                              short dstLength)
                       throws java.lang.NullPointerException,
                              java.lang.ArrayIndexOutOfBoundsException,
                              ToolkitException
```

6.2.4.15.1.1 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.

6.2.4.15.1.2 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.

6.2.4.15.1.3 Context errors

CRRC1: if the handler is busy an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException HANDLER_NOT_AVAILABLE.

6.2.4.15.2 Test Suite files

Test Script:	API_2_ENH_FACYBS_BSS_1.scr
Test Applet:	~~API_2_ENH_FACYBS_BSS_1.java~~
~~Load Script:~~	~~API_2_ENH_FACYBS_BSS_1.ldr~~
~~Cleanup Script:~~	~~API_2_ENH_FACYBS_BSS_1.clr~~
~~Parameter File:~~	API_2_ENH_FACYBS_BSS_1.java
Load Script:	API_2_ENH_FACYBS_BSS_1.ldr
Cleanup Script:	API_2_ENH_FACYBS_BSS_1.clr
Parameter File:	API_2_ENH_FACYBS_BSS_1.par

6.2.4.15.3 Test procedure

Id	Description	API Expectation	APDU Expectation
	Fill the SMS PP with TLV: 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**
	tag = 06h, occurrence = 1
	valueOffset = 0
	dstBuffer.length = 5
	dstOffset = 5
	dstLength = 1
		ArrayIndexOutOfBoundsException is thrown	

3	**dstOffset < 0**
	dstBuffer.length = 5
	dstOffset = -1
	dstLength = 1
		ArrayIndexOutOfBoundsException is thrown	

4	**dstLength >dstBuffer.length**
	dstBuffer.length = 5
	dstOffset = 0
	dstLength = 6
		ArrayIndexOutOfBoundsException is thrown	

5	**dstOffset + dstLength >dstBuffer.length**
	dstBuffer.length = 5
	dstOffset = 3
	dstLength = 3
		ArrayIndexOutOfBoundsException is thrown	

6	**dstLength < 0**
	dstBuffer.length = 5
	dstOffset = 0
	dstLength = -1
		ArrayIndexOutOfBoundsException is thrown	

7	**valueOffset ≥ Value Length**
	tag = 06h, occurrence = 1
	valueOffset = 6
	dstBuffer.length = 15
	dstOffset = 0
	dstLength = 1
		ToolkitException.OUT_OF_TLV_BOUNDARIES is thrown	

8	**valueOffset < 0**
	valueOffset = -1
	dstBuffer.length = 15
	dstOffset = 0
	dstLength = 1
		ToolkitException.OUT_OF_TLV_BOUNDARIES is thrown	

9	**dstLength > Value length**
	valueOffset = 0
	dstBuffer.length = 15
	dstOffset = 0
	dstLength = 7
		ToolkitException.OUT_OF_TLV_BOUNDARIES is thrown	

10	**valueOffset + dstLength > Text String** length
	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	

	~~**Call the getValueLength() method**~~	~~ToolkitException.UNAVAILABLE_ELEMENT is thrown.~~	
12	**Call the getValueLength() method**	ToolkitException.UNAVAILABLE_ELEMENT is thrown.	
~~12~~	~~**Successful call**~~ ~~tag = 06h, occurrence = 1~~ ~~valueOffset = 0~~ ~~dstBuffer.length = 06~~ ~~dstOffset = 0~~ ~~dstLength = 06~~	~~Result of findAndCopyValue() is 6~~	

13	**Successful call**
	tag = 06h, occurrence = 1
	valueOffset = 0
	dstBuffer.length = 06
	dstOffset = 0
	dstLength = 06
		Result of findAndCopyValue() is 6	

~~13~~	~~**Compare buffer**~~ ~~buffer = 81 11 22 33 44 F5~~	~~Result is 00h~~	
14	**Compare buffer** buffer = 81 11 22 33 44 F5	Result is 00h	
~~14~~	~~**initialize dstBuffer**~~ ~~dstBuffer = 55 55 … 55~~		
15	**initialise dstBuffer** dstBuffer = 55 55 … 55		
	Successful call tag = 06h, occurrence = 1	Result of findAndCopyValue () is 0007	

	valueOffset = 2
	dstBuffer.length = 12
	dstOffset = 3
	dstLength = 04		

~~15~~	~~**Compare buffer**~~ ~~buffer =~~ ~~55 55 55 22 33 44 F5 55 55 55 55 55~~	~~Result is 00h~~	
16	**Compare buffer** buffer = 55 55 55 22 33 44 F5 55 55 55 55 55	Result is 00h	
~~16~~	~~**Successful call**~~ ~~tag = 02h, occurrence = 1~~ ~~valueOffset = 0~~ ~~dstBuffer.length = 12~~ ~~dstOffset = 0~~ ~~dstLength = 2~~	~~Result of findAndCopyValue() is 0002~~	

17	**Successful call**
	tag = 02h, occurrence = 1
	valueOffset = 0
	dstBuffer.length = 12
	dstOffset = 0
	dstLength = 2
		Result of findAndCopyValue() is 0002	

~~17~~	~~**Compare buffer**~~ ~~buffer = 83 81 55 … 55~~	~~Result is 00h~~	
18	**Compare buffer** buffer = 83 81 55 … 55	Result is 00h	
~~18~~	~~**Successful call**~~ ~~tag = 02h, occurrence = 2~~ ~~valueOffset = 0~~ ~~dstBuffer.length = 12~~ ~~dstOffset = 0~~ ~~dstLength = 2~~	~~Result of findAndCopyValue() is 0002~~	

19	**Successful call**
	tag = 02h, occurrence = 2
	valueOffset = 0
	dstBuffer.length = 12
	dstOffset = 0
	dstLength = 2
		Result of findAndCopyValue() is 0002	

~~19~~	~~**Compare buffer**~~ ~~buffer = 22 44 55 … 55~~	~~Result is 00h~~	
20	**Compare buffer** buffer = 22 44 55 … 55	Result is 00h	
~~20~~	~~**Successful call (with tag 82h)**~~ ~~tag = 82h~~ ~~occurrence = 1~~ ~~valueOffset = 0~~ ~~dstBuffer.length = 12~~ ~~dstOffset = 0~~ ~~dstLength = 02~~	~~Result of findAndCopyValue () is 0002~~	

21	**Successful call (with tag 82h)**
	tag = 82h
	occurrence = 1
	valueOffset = 0
	dstBuffer.length = 12
	dstOffset = 0
	dstLength = 02
		Result of findAndCopyValue () is 0002	

~~21~~	~~**Compare buffer**~~ ~~buffer = 83 81 55 … 55~~	~~Result is 00h~~	
22	**Compare buffer** buffer = 83 81 55 … 55	Result is 00h	
~~22~~	~~**Successful call (with tag 82h)**~~ ~~tag = 82h~~ ~~occurrence = 2~~ ~~valueOffset = 0~~ ~~dstBuffer.length = 12~~ ~~dstOffset = 0~~ ~~dstLength = 02~~	~~Result of findAndCopyValue () is 0002~~	

23	**Successful call (with tag 82h)**
	tag = 82h
	occurrence = 2
	valueOffset = 0
	dstBuffer.length = 12
	dstOffset = 0
	dstLength = 02
		Result of findAndCopyValue () is 0002	

~~23~~	~~**Compare buffer**~~	~~Result is 00h~~	

	~~Buffer = 22 44 55 … 55~~		
24	**Compare buffer** `Buffer = 22 44 55 … 55`	Result is 00h	
~~24~~	~~**Successful call, findAndCopyValue with length =0**~~ ~~`DstBuffer.length = 12`~~ ~~`dstOffset = 12`~~ ~~`dstLength = 0`~~	~~Result of findAndCopyValue () is 12~~	

25	**Successful call, findAndCopyValue with length =0**
	`DstBuffer.length = 12`
	`dstOffset = 12`
	`dstLength = 0`
		Result of findAndCopyValue () is 12	

	Send Formatted SMS PP with the maximum user data length = 0x010D, using 2 concatenated envelopes		

26	**Successful call**
	`tag = 0Bh, occurrence = 1`
	`valueOffset = 0x11`
	`dstBuffer.length = 0x010D`
	`dstOffset = 0`
	`dstLength = 0x010D`
		Result of findAndCopyValue() is 0x010D	

27	**Compare buffer** `buffer = 0348 Header + secured data (01 02 … FA)`	Result is 00h	
28	**initialise dstBuffer** `dstBuffer = 55 55 … 55`		

	Successful call
	`tag = 0Bh, occurrence = 1`
	`valueOffset = 0x0111`
	`dstBuffer.length = 0x010D`
	`dstOffset = 0x0100`
	`dstLength = 0x0D`
		Result of findAndCopyValue () is 0x010D	

29	**Compare buffer**
	`buffer =`
	`55 55 … 55 55 EE EF F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA`
		Result is 00h	

6.2.4.15.4 Test Coverage

CRR number	Test case number
~~N1~~	~~13, 15, 17, 19~~
~~N2~~	~~11~~
~~N3~~	~~12, 14, 16, 18, 24~~
~~N4~~	~~20, 21, 22, 23~~
N1	14, 15, 17, 19, 20
N2	11, 12
N3	13, 15, 17, 19, 25
N4	21, 22, 23, 24, 26, 27, 28,29
P1	1
P2	2, 3, 4, 5, 6
P3	7, 8, 9, 10
C1	Does not apply for EnvelopeHandler

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

Test Area Reference: API_2_ENH_FACRB_BS~~.~~

6.2.4.16.1 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
```

6.2.4.16.1.1 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 simple TLV is less than that in compareBuffer returns -1.

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

CRRN6: The search method is comprehension required flag independent.

6.2.4.16.1.2 Parameter errors

CRRP1: if compareBuffer is null NullPointerException shall be thrown.

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

6.2.4.16.1.3 Context errors

CRRC1: if the handler is busy an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException HANDLER_NOT_AVAILABLE.

6.2.4.16.2 Test Suite files

Test Script: API_2_ENH_FACRB_BS_1.scr

Test Applet: ~~API_2_ENH_FACRB_BS_1.java~~

~~Load Script: API_2_ENH_FACRB_BS_1.ldr~~

~~Cleanup Script: API_2_ENH_FACRB_BS_1.clr~~

~~Parameter File:~~ API_2_ENH_FACRB_BS_1.java

Load Script: API_2_ENH_FACRB_BS_1.ldr

Cleanup Script: API_2_ENH_FACRB_BS_1.clr

Parameter File: API_2_ENH_FACRB_BS_1.par

6.2.4.16.3 Test procedure

Id	Description	API Expectation	APDU Expectation
	Fill the SMS PP with TLV: 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**
	`tag = 06h`
	`compareBuffer.length = 12`
	`compareOffset = 12`
		ArrayIndexOutOfBoundsException is thrown	

3	**compareOffset < 0**
	`compareBuffer.length = 12`
	`compareOffset = -1`
		ArrayIndexOutOfBoundsException is thrown	

4	**length > compareBuffer.length**
	`compareBuffer.length = 05`
	`compareOffset = 0`
		ArrayIndexOutOfBoundsException is thrown	

5	**compareOffset + length > compareBuffer.length**
	`compareBuffer.length = 12`
	`compareOffset = 7`
		ArrayIndexOutOfBoundsException is thrown	

6	**Select a TLV (tag 02h)**		

	findAndCompareValue()
	`tag = 03h`
		ToolkitException.UNAVAILABLE_ELEMENT is thrown	

	~~**Call the getValueLength() method**~~	~~ToolkitException.UNAVAILABLE_ELEMENT is thrown.~~	
7	**Call the getValueLength() method**	ToolkitException.UNAVAILABLE_ELEMENT is thrown.	
~~7~~	~~**Initialize compareBuffer**~~ ~~`compareBuffer = 81 11 22 33 44 F5`~~		
8	**Initialise compareBuffer** `compareBuffer = 81 11 22 33 44 F5`		

	Compare buffers
	`tag = 06h`
	`compareOffset = 0`
		Result is 00h	

~~8~~	~~**Verify current TLV**~~ ~~`getValueLength()`~~	~~Result is 06~~	
9	**Verify current TLV** `getValueLength()`	Result is 06	
~~9~~	~~**Initialize compareBuffer**~~ ~~`compareBuffer = 81 11 22 33 44 F4`~~		
10	**Initialise compareBuffer** `compareBuffer = 81 11 22 33 44 F4`		
	Compare buffers with same parameters	Result is +1	
~~10~~	~~**Initialize compareBuffer**~~ ~~`compareBuffer = 81 11 22 33 44 F6`~~		
11	**Initialise compareBuffer** `compareBuffer = 81 11 22 33 44 F6`		
	Compare buffers with same parameters	Result is -1	
~~11~~	~~**Initialize compareBuffer**~~ ~~`compareBuffer =`~~ ~~`55 55 81 11 22 33 44 F5 55 55 55 55`~~		

12	**Initialise compareBuffer**
	`compareBuffer =`
	`55 55 81 11 22 33 44 F5 55 55 55 55`		

	Compare buffers `compareOffset = 2`	Result is 00h	
~~12~~	~~**Initialize compareBuffer**~~ ~~`compareBuffer =`~~ ~~`55 55 83 81 55 55 55 55 55 55 55 55`~~		

13	**Initialise compareBuffer**
	`compareBuffer =`
	`55 55 83 81 55 55 55 55 55 55 55 55`		

	Compare buffers `compareOffset = 2`	Result is 00h	
~~13~~	~~**Initialize compareBuffer**~~ ~~`compareBuffer =`~~ ~~`55 55 83 80 55 55 55 55 55 55 55 55`~~		

14	**Initialise compareBuffer**
	`compareBuffer =`
	`55 55 83 80 55 55 55 55 55 55 55 55`		

	Compare buffers `compareOffset = 2`	Result is +1	
~~14~~	~~**Initialize compareBuffer**~~ ~~`compareBuffer =`~~ ~~`55 55 83 82 55 55 55 55 55 55 55 55`~~		

15	**Initialise compareBuffer**
	`compareBuffer =`
	`55 55 83 82 55 55 55 55 55 55 55 55`		

	Compare buffers `compareOffset = 2`	Result is –1	
~~15~~	~~**Initialize compareBuffer**~~ ~~`compareBuffer =`~~ ~~`83 81 55 55 55 55 55 55 55 55 55 55`~~		

16	**Initialise compareBuffer**
	`compareBuffer =`
	`83 81 55 55 55 55 55 55 55 55 55 55`		

	Successful call (with tag 02h) `tag = 02h`	Result is 00h	

	compareBuffer.length = 12
	compareOffset = 0		

~~16~~	~~**Initialize compareBuffer**~~ ~~CompareBuffer = 01 02 … C4~~		
17	**Initialise compareBuffer** CompareBuffer = 01 02 … C4		

	Successful call (with tag B3h)
	Tag = B3h
	CompareBuffer.length = C4
	CompareOffset = 0
		Result is 00h	

	Send Unformatted SMS PP with the maximum user data length = 0x010C, using 2 concatenated envelopes		
	Initialise compareBuffer CompareBuffer = 0340 Header + user data (00 01 02 … FF 01 … 0C)		

18	**Successful call (with SMS TPDU TAG)**
	Tag = 0Bh
	CompareBuffer.length = 0x011E
	CompareOffset = 0
		Result is 00h	

	Initialise compareBuffer CompareBuffer = 55 55 … 55 CompareBuffer from offset 0x0100= 0340 Header + user data (00 01 02 … FF 01 … 0C)		

19	**Successful call (with SMS TPDU TAG)**
	Tag = 0Bh
	CompareBuffer.length = 0x220
	CompareOffset = 0x0100
		Result is 00h	

6.2.4.16.4 Test Coverage

CRR number	Test case number
~~N1~~	~~6~~
~~N2~~	~~8~~
~~N3~~	~~7, 11, 12~~
~~N4~~	~~9, 13~~
~~N5~~	~~10, 14~~
~~N6~~	~~15, 16~~
N1	6,7
N2	9
N3	8, 12, 13, 18, 19
N4	10, 14
N5	11, 15
N6	16, 17
P1	1
P2	2, 3, 4, 5
C1	Does not apply for EnvelopeHandler

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

Test Area Reference: API_2_ENH_FACRBBS_BSS~~.~~

6.2.4.17.1 Conformance requirement

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

```
public byte findAndCompareValue(byte tag,
                                byte occurence,
                                short valueOffset,
                                byte[] compareBuffer,
                                short compareOffset,
                                short compareLength)
                         throws java.lang.NullPointerException,
~~java.lang.ArrayIndexOutOfBoundsException,~~
~~                                ToolkitException~~
                                java.lang.ArrayIndexOutOfBoundsException,
                                ToolkitException
```

6.2.4.17.1.1 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 simple TLV is less than that in compareBuffer -1 is returned.

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

CRRN6: The search method is comprehension required flag independent.

6.2.4.17.1.2 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. occurence = 0) an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException BAD_INPUT_PARAMETER.

6.2.4.17.1.3 Context errors

CRRC1: if the handler is busy an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException HANDLER_NOT_AVAILABLE.

6.2.4.17.2 Test Suite files

Test Script:	API_2_ENH_FACRBBS_BSS_1.scr
Test Applet:	~~API_2_ENH_FACRBBS_BSS_1.java~~
~~Load Script:~~	~~API_2_ENH_FACRBBS_BSS_1.ldr~~
~~Cleanup Script:~~	~~API_2_ENH_FACRBBS_BSS_1.clr~~
~~Parameter File:~~	API_2_ENH_FACRBBS_BSS_1.java
Load Script:	API_2_ENH_FACRBBS_BSS_1.ldr
Cleanup Script:	API_2_ENH_FACRBBS_BSS_1.clr
Parameter File:	API_2_ENH_FACRBBS_BSS_1.par

6.2.4.17.3 Test procedure

Id	Description	API Expectation	APDU Expectation
	Fill the SMS PP with TLV: 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**	ArrayIndexOutOfBoundsExceptio	

	`tag = 06h, occurrence = 1`
	`valueOffset = 0`
	`compareBuffer.length = 6`
	`compareOffset = 6`
	`compareLength = 1`
		n is thrown	

3	**compareOffset < 0**
	`compareBuffer.length = 6`
	`compareOffset = -1`
	`compareLength = 1`
		ArrayIndexOutOfBoundsException is thrown	

4	**compareLength >compareBuffer.length**
	`compareBuffer.length = 5`
	`compareOffset = 0`
	`compareLength = 6`
		ArrayIndexOutOfBoundsException is thrown	

5	**compareOffset + compareLength >compareBuffer.length**
	`compareBuffer.length = 5`
	`compareOffset = 3`
	`compareLength = 3`
		ArrayIndexOutOfBoundsException is thrown	

6	**compareLength < 0**
	`compareBuffer.length = 5`
	`compareOffset = 0`
	`compareLength = -1`
		ArrayIndexOutOfBoundsException is thrown	

7	**valueOffset ≥ Value Length**
	`tag = 06h, occurrence = 1`
	`valueOffset = 6`
	`compareBuffer.length = 15`
	`compareOffset = 0`
	`compareLength = 1`
		ToolkitException.OUT_OF_TLV_BOUNDARIES is thrown	

8	**valueOffset < 0**
	`valueOffset = -1`
	`compareBuffer.length = 15`
	`compareOffset = 0`
	`compareLength = 1`
		ToolkitException.OUT_OF_TLV_BOUNDARIES is thrown	

9	**compareLength > Value length**
	`valueOffset = 0`
	`compareBuffer.length = 15`
	`compareOffset = 0`
	`compareLength = 7`
		ToolkitException.OUT_OF_TLV_BOUNDARIES is thrown	

10	**valueOffset + compareLength > Value length**
	`valueOffset = 2`
	`compareBuffer.length = 15`
	`compareOffset = 0`
	`compareLength = 5`
		ToolkitException.OUT_OF_TLV_BOUNDARIES is thrown	

~~11~~	~~**Invalid parameter**~~ ~~`occurrence = 0`~~	~~ToolkitException.BAD_INPUT_PARAMETER is thrown~~	
11	**Invalid parameter** `occurrence = 0`	ToolkitException.BAD_INPUT_PARAMETER is thrown	
12	**Select a TLV (tag 02h)**		
	~~**findAndCompareValue()**~~ ~~`tag = 06h`~~ ~~`occurrence = 2`~~	~~ToolkitException.UNAVAILABLE_ELEMENT is thrown~~	

	findAndCompareValue()
	`tag = 06h`
	`occurrence = 2`
		ToolkitException.UNAVAILABLE_ELEMENT is thrown	

	~~**Call the getValueLength() method**~~	~~ToolkitException.UNAVAILABLE_ELEMENT is thrown.~~	
13	**Call the getValueLength() method**	ToolkitException.UNAVAILABLE_ELEMENT is thrown.	
~~13~~	~~**Initialize compareBuffer**~~ ~~`compareBuffer = 81 11 22 33 44 F5`~~		
14	**Initialise compareBuffer** `compareBuffer = 81 11 22 33 44 F5`		

	findAndCompareValue()
	`tag = 06h, occurrence = 1`
	`valueOffset = 0`
	`compareOffset = 0`
	`compareLength = 6`
		Result is 00h	

~~14~~	~~**Verify current TLV**~~ ~~`getValueLength()`~~	~~Result is 0006~~	
15	**Verify current TLV** `getValueLength()`	Result is 0006	
~~15~~	~~**Initialize compareBuffer**~~ ~~`compareBuffer = 81 11 22 33 44 F4`~~		

16	**Initialise compareBuffer** `compareBuffer = 81 11 22 33 44 F4`		
	Compare buffers with same parameters	Result is +1	
~~16~~	~~**Initialize compareBuffer**~~ ~~`compareBuffer = 81 11 22 33 44 F6`~~		
17	**Initialise compareBuffer** `compareBuffer = 81 11 22 33 44 F6`		
	Compare buffers with same parameters	Result is -1	
~~17~~	~~**Initialize compareBuffer**~~ ~~`compareBuffer =`~~ ~~`55 55 55 22 33 44 F5 55 55 55 55`~~		

18	**Initialise compareBuffer**
	`compareBuffer =`
	`55 55 55 22 33 44 F5 55 55 55 55`		

	Compare buffers
	`valueOffset = 2`
	`compareOffset = 3`
	`compareLength = 4`
		Result is 00h	

~~18~~	~~**Initialize compareBuffer**~~ ~~`compareBuffer =`~~ ~~`55 55 55 22 33 45 F5 55 55 55 55`~~		

19	**Initialise compareBuffer**
	`compareBuffer =`
	`55 55 55 22 33 45 F5 55 55 55 55`		

	Compare buffers with same parameters	Result is -1	
~~19~~	~~**Initialize compareBuffer**~~ ~~`compareBuffer =`~~ ~~`55 55 55 22 33 43 F5 55 55 55 55`~~		

20	**Initialise compareBuffer**
	`compareBuffer =`
	`55 55 55 22 33 43 F5 55 55 55 55`		

	Compare buffers with same parameters	Result is +1	
~~20~~	~~**Initialize compareBuffer**~~ ~~`compareBuffer =`~~ ~~`83 81 55 55 55 55 55 55 55 55 55 55`~~		

21	**Initialise 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	

~~21~~	~~**Initialize compareBuffer**~~ ~~`compareBuffer =`~~ ~~`22 44 55 55 55 55 55 55 55 55 55 55`~~		

22	**Initialise 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	

~~22~~	~~**Initialize compareBuffer**~~ ~~`compareBuffer =`~~ ~~`22 45 55 55 55 55 55 55 55 55 55 55`~~		

23	**Initialise 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	

~~23~~	~~**Initialize compareBuffer**~~ ~~`compareBuffer =`~~ ~~`83 81 55 55 55 55 55 55 55 55 55 55`~~		

24	**Initialise compareBuffer**
	`compareBuffer =`
	`83 81 55 55 55 55 55 55 55 55 55 55`		

	Successful call (with tag 02h) `tag = 02h, occurrence = 1`	Result is 00h	

	valueOffset = 0
	compareBuffer.length = 12
	compareOffset = 0
	compareLength = 2		

~~24~~	~~**Initialize compareBuffer**~~ ~~compareBuffer = 01 02 … C4~~		
25	**Initialise compareBuffer** compareBuffer = 01 02 … C4		

	Successful call (with tag B3h)
	tag = B3h, occurrence = 1
	valueOffset = 0
	compareBuffer.length = 00C4
	compareOffset = 0
	compareLength = 00C4
		Result is 00h	

~~25~~	~~**Successful call, findAndCompareValue with length =0**~~ ~~DstBuffer.length = C4~~ ~~DstOffset = C4~~ ~~DstLength = 0~~	~~Result of findAndCompareValue() is 00h~~	

26	**Successful call, findAndCompareValue with length =0**
	DstBuffer.length = C4
	DstOffset = C4
	DstLength = 0
		Result of findAndCompareValue() is 00h	

	Send Formatted SMS PP with the maximum user data length = 0x010D, using 2 concatenated envelopes		
	Initialise compareBuffer CompareBuffer = 23.048 Header + secured data (01 02 … FA)		

27	**Successful call (with SMS TPDU TAG)**
	tag = 0Bh, occurrence = 1
	valueOffset = 0x11
	compareBuffer.length = 0x010D
	compareOffset = 0
	compareLength = 0x010D
		Result is 00h	

	Initialise compareBuffer CompareBuffer = 55 55 … 55 EE EF F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA		

28	**Successful call (with SMS TPDU TAG)**
	tag = 0Bh, occurrence = 1
	valueOffset = 0x11
	compareBuffer.length = 0x010D
	compareOffset = 0x0100
	compareLength = 0x0D
		Result is 00h	

6.2.4.17.4 Test Coverage

CRR number	Test case number
~~N1~~	~~12~~
~~N2~~	~~14~~
~~N3~~	~~13, 17, 20, 21, 25~~
~~N4~~	~~19, 15~~
~~N5~~	~~16, 18, 22~~
~~N6~~	~~23, 24~~
N1	12, 13
N2	15
N3	14, 18, 21, 22, 26, 27, 28
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 EnvelopeHandler

6.2.4.18 Method getCapacity

Test Area Reference: API_2_ENH_GCAP

6.2.4.18.1 Conformance Requirement:

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

```
public byte getCapacity()
```

6.2.4.18.1.1 Normal execution

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

6.2.4.18.2 Test suite files

Test Script:	API_2_ENH_GCAP_1.scr
Test Applet:	API_2_ENH_GCAP_1.java
Load Script:	API_2_ENH_GCAP_1.ldr
Cleanup Script:	API_2_ENH_GCAP_1.clr
Parameter File:	API_2_ENH_GCAP_1.par

6.2.4.18.3 Test Procedure

Id	Description	API Expectation	APDU Expectation
1	**EnvelopeHandler available** 1 - Send envelope SMS-PP Formatted 2 - The applet calls the getLength() method 3 - The applet calls 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	

6.2.4.18.4 Test Coverage

CRR number	Test case number
N1	1

6.2.4.19 Method getUserDataLength

Test Area Reference: API_2_ENH_GUDL

6.2.4.19.1 Conformance Requirement:

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

```
public short getUserDataLength()
```

6.2.4.19.1.1 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_UNFORMATTED_SMS_PP_ENV.

CRRN7: The method can be used if the event is EVENT_UNFORMATTED_SMS_PP_UDP.

6.2.4.19.1.2 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.

6.2.4.19.2 Test suite files

Specific triggering:

- UNFORMATTED_SMS_PP_ENV
- FORMATTED_SMS_PP_UPD
- UNFORMATTED_SMS_PP_UPD
- UNRECOGNIZED_ENVELOPE
- For Formatted triggering if CC/RC/DS is used, the security parameters are those used for downloading applications.

Test Script:	API_2_ENH_GUDL_1.scr
Test Applet:	API_2_ENH_GUDL_1.java
Load Script:	API_2_ENH_GUDL_1.ldr
Cleanup Script:	API_2_ENH_GUDL_1.clr
Parameter File:	API_2_ENH_GUDL_1.par

6.2.4.19.3 Test Procedure

Id	Description	API 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 0x05	Returns 0x001F	
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.	GetValueByte() returns 0x40(23.040 first byte)	

	GetUserDataLength() and then getValueByte() with offset 0		
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 0x05	Returns 0x001F	
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(23.040 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 (23.040 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 (23.040 first byte)	
	UNRECOGNIZED_ENVELOPE Triggering		
47	Test with an UNRECOGNIZED_ENVELOPE	ToolkitException UNAVAILABLE_ELEMENT	

6.2.4.19.4 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

6.2.4.20 Method getChannelIdentifier

Test Area Reference: API_2_ENH_GCID

6.2.4.20.1 Conformance Requirement:

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

```
public byte getChannelIdentifier()
                                    throws ToolkitException
```

6.2.4.20.1.1 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 EnvelopeHandler is available.

6.2.4.20.1.2 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 Simple TLV Channel Status length is equal to 0.

6.2.4.20.2 Test suite files

Test Script:	API_2_ENH_GCID_1.scr
Test Applet:	API_2_ENH_GCID_1.java
Load Script:	API_2_ENH_GCID_1.ldr
Cleanup Script:	API_2_ENH_GCID_1.clr

Parameter File: API_2_ENH_GCID_1.par

6.2.4.20.3 Test Procedure

Id	Description	API 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() methods are called.		2- OPEN CHANNEL proactive command is fetched TERMINAL RESPONSE is issued with Channel Id from 01 to 07
1	1- Send envelope Event Download Channel Status with channel status TLV: channel status value = 0x8100. 2- Call EnvelopeHandler.getChannelIdentifier() method	1- Applet1 is triggered 2- Returns 0x01	
2	1- Send envelope Event Download Channel Status with two channel status TLV: first value = 0x8400 second value = 0x8500. 2- Call twice the EnvelopeHandler.getChannelIdentifier() method	2- Returns 0x04 Returns 0x04	
3	1- Send envelope Event Download Channel Status with channel status TLV: Channel Status value = 0x0605 ViewHandler.FindTLV() with Device IdentityTag. 2- Call EnvelopeHandler.getChannelIdentifier() method. 3- Compare EnvelopeHandler.getChannelIdentifier() and then ViewHandler.getValueByte(0).	2- Returns 0x06 3- GetChannelIdentifier() =getValueByte(0)	
4	1- Send envelope Menu Selection without Channel Status TLV. 2- Call EnvelopeHandler.getChannelIdentifier() method.	2- A Toolkit exception UNAVAILABLE_ELEMENT is thrown.	
5	1- Send Envelope Event Download Channel Status with Channel Status TLV: Channel status value = 0x0600 2- Call EnvelopeHandler.getChannelIdentifier() method.	1- Returns 0x06	
6	1- Send unrecognized envelope with a Channel Status TLV having a length equal to 0. 2- Call EnvelopeHandler.getChannelIdentifier() method.	2- A Toolkit exception OUT_OF_TLV_BOUNDARIES is thrown.	

6.2.4.20.4 Test Coverage

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

6.2.5 Class EnvelopeResponseHandler

6.2.5.21 Method getCapacity

Test Area Reference: API_2_ERH_GCAP

6.2.5.21.1 Conformance Requirement:

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

```
Public byte getCapacity()
```

6.2.5.21.1.1 Normal execution

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

6.2.5.21.1.2 Context errors

CRRC1: The method shall throw HANDLER_NOT_AVAILABLE ToolkitException if the handler is busy.

6.2.5.21.2 Test suite files

Test Script:	API_2_ERH_GCAP_1.scr
Test Applet:	API_2_ERH_GCAP_1.java
Load Script:	API_2_ERH_GCAP_1.ldr
Cleanup Script:	API_2_ERH_GCAP_1.clr
Parameter File:	API_2_ERH_GCAP_1.par

6.2.5.21.3 Test Procedure

Id	Description	API Expectation	APDU Expectation
1	**EnvelopeResponseHandler available** 1- Send envelope SMS-PP Formatted 2- The applet calls getTheHandler() method 3- The applet calls getCapacity() method on the EnvelopeResponseHandler 4- The applet fills the handler with the maximum capacity using AppendTLV() method 5- The applet calls clear() method on the EnvelopeResponseHandler 6- The applet fills the handler with the maximum capacity plus one, using AppendTLV() method	1- Applet is triggered 2- No exception is thrown 3- No exception is thrown 4- No exception is thrown 5- No exception is thrown 6- HANDLER_OVERFLOW exception is thrown	

6.2.5.21.4 Test Coverage

CRR number	Test case number
N1	1
C1	Tested in Framework part: FWK_MHA_ERHD

6.2.6 Class ProactiveHandler

6.2.7.3 Method initDisplayText

Test Area Reference: API_2_PAH_INDTBB_BSS~~.~~

6.2.7.3.1 Conformance requirement

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

```
public void initDisplayText(byte qualifier,
                            byte dcs,
                            byte[] buffer,
                            short offset,
                            short length)
                     throws java.lang.NullPointerException,
                            java.lang.ArrayIndexOutOfBoundsException,
                            ToolkitException
```

6.2.7.3.1.1 Normal execution

CRRN1: The method shall build a DISPLAY TEXT proactive command in the ProactiveHandler, using qualifier, dcs and buffer parameters. Comprehension required flags are set.

~~-~~CRRN2: A call to this method clears the handler then ~~initializes~~initialises it.

CRRN3: No TLV is selected after a call to the method.

CRRN4: The DISPLAY TEXT command is not sent by the method.

CRRN5: The Command Number may take any value between 01h and FEh.

CRRN6: If length is equal to zero, then the Text String TLV inserted in the command is a null text string TLV as defined in TS 11.14 [4].

6.2.7.3.1.2 Parameter errors

CRRP1: The method shall throw NullPointerException if buffer is null.

CRRP2: If offset or length or both would cause access outside array bounds, an ArrayIndexOutOfBoundsException shall be thrown.

6.2.7.3.1.3 Context errors

CRRC1: A ToolkitException.HANDLER_OVERFLOW shall be thrown if the ProactiveHandler is too small to put the requested data.

6.2.7.3.2 Test Suite files

Test Script:	API_2_PAH_INDTBB_BSS_1.scr
Test Applet:	~~API_2_PAH_INDTBB_BSS_1.java~~
~~Load Script:~~	~~API_2_PAH_INDTBB_BSS_1.ldr~~
~~Cleanup Script:~~	~~API_2_PAH_INDTBB_BSS_1.clr~~
~~Parameter File:~~	API_2_PAH_INDTBB_BSS_1.java
Load Script:	API_2_PAH_INDTBB_BSS_1.ldr
Cleanup Script:	API_2_PAH_INDTBB_BSS_1.clr
Parameter File:	API_2_PAH_INDTBB_BSS_1.par

6.2.7.3.3 Test procedure

Id	Description	API Expectation	APDU Expectation
1	**NULL as parameter to buffer** `buffer = NULL`	NullPointerException is thrown	

2	**offset > buffer.length**
	`buffer = "Text"`
	`offset = 5`
	`length = 0`
		ArrayIndexOutOfBoundsException is thrown	

3	**offset < 0**
	`buffer = "Text"`
	`offset = -1`
		ArrayIndexOutOfBoundsException is thrown	

4	**length > buffer.length**
	`buffer = "Text"`
	`offset = 0`
	`length = 5`
		ArrayIndexOutOfBoundsException is thrown	

5	**offset + length > buffer.length**
	`buffer = "Text"`
	`offset = 3`
	`length = 2`
		ArrayIndexOutOfBoundsException is thrown	

6	**length < 0**
	`buffer = "Text"`
	`offset = 3`
	`length = -1`
		ArrayIndexOutOfBoundsException is thrown	

7	**Successful call, buffer is the whole buffer**
	`qualifier = 0`
	`dcs = 4`
	`buffer = "TextA"`
	`offset = 0`
	`length = 5`
		No exception is thrown	

	Verify the command number value	Command number between 01h and FEh	

8	**Send the command**		DISPLAY TEXT Proactive command
			
			`qualifier = 00h`
			`dcs = 4`
			`Text = "TextA"`

9	**Successful call, buffer is part of a buffer with the end part**
	Send the command
	`qualifier = 0`
	`dcs = 4`
	`buffer = "12TextB"`
	`offset = 2`
	`length = 5`
			DISPLAY TEXT Proactive command
			
			`qualifier = 00h`
			`dcs = 4`
			`Text = "TextB"`

10	**Successful call, buffer is part of a buffer with the first part**
	Send the command
	`qualifier = 0`
	`dcs = 4`
	`buffer = "TextC12"`
	`offset = 0`
	`length = 5`
			DISPLAY TEXT Proactive command
			
			`qualifier = 00h`
			`dcs = 4`
			`Text = "TextC"`

11	**Successful call, buffer is part of a buffer**
	Send the command
	`qualifier = 0`
	`dcs = 4`
	`buffer = "12TextD34"`
	`offset = 2`
	`length = 5`
			DISPLAY TEXT Proactive command
			
			`qualifier = 00h`
			`dcs = 4`
			`Text = "TextD"`

12	**Successful call, qualifier = 81h**
	Send the command
	`qualifier = 81h`
	`dcs = 4`
	`buffer = "TextE"`
	`offset = 0`
	`length = 5`
			DISPLAY TEXT Proactive command
			
			`qualifier = 81h`
			`dcs = 4`
			`Text = "TextE"`

13	**Successful call, DCS=0 (7 bits)**
	Send the command
	`qualifier = 0`
	`dcs = 0`
	`buffer = "TextF"`
	`offset = 0`
			DISPLAY TEXT Proactive command
			
			`qualifier = 00h`
			`dcs = 0`
			`Text = "TextF"`

	`length = 5`		

14	**Successfull call, DCS=8 (UCS2)**
	Send the command
	`qualifier = 0`
	`dcs = 8`
	`buffer = "TextG"`
	`offset = 0`
	`length = 5`
			DISPLAY TEXT Proactive command
			
			`qualifier = 00h`
			`dcs = 8`
			`Text = "TextG"`

15	**Call the initDisplayText() method with any value**
	Then build and send a DISPLAY TEXT command
	`qualifier = 0`
	`dcs = 4`
	`buffer = "TextHTextH"`
	`offset = 0`
	`length = 10`
			DISPLAY TEXT Proactive command
			
			`qualifier = 00h`
			`dcs = 4`
			`Text = "TextHTextH"`

~~16~~	~~**Successful call, text length is null**~~ ~~**Send the command**~~ ~~`qualifier = 0`~~ ~~`dcs = 4`~~ ~~`buffer = "" (not null buffer)`~~ ~~`offset = 0`~~ ~~`length = 0`~~		~~DISPLAY TEXT Proactive command~~ ~~`qualifier = 00h`~~ ~~`Text String TLV = 8D 00`~~

16	**Successful call, text length is zero**
	Send the command
	`qualifier = 0`
	`dcs = 4`
	`buffer = "TextHTextH"`
	`offset = 0`
	`length = 0`
			DISPLAY TEXT Proactive command
			
			`qualifier = 00h`
			`Text String TLV = 8D 00`

17	**Select a TLV in the ProactiveHandler** **Call the initDisplayText() method** **Call the getValueLength() method**	UNAVAILABLE_ELEMENT ToolkitException is thrown by getValueLength()	

18	**Successful call, buffer length = 7Eh**
	
	`qualifier = 0`
	`dcs = 4`
	`buffer = "UUU…"`
	`offset = 0`
	`length = 7Eh`
			DISPLAY TEXT Proactive command
			
			`Text String TLV =`
			`8D 7F 04 55 55…`

19	**Successful call, buffer length = 7Fh**
	
	`qualifier = 0`
	`dcs = 4`
	`buffer = "UUU…"`
	`offset = 0`
	`length = 7Fh`
			DISPLAY TEXT Proactive command
			
			`Text String TLV = 8D 81`
			`80 04 55 55…`

20	**Successful call, buffer length = 240**
	
	`Qualifier = 0`
	`dcs = 4`
	`buffer = "UUU…"`
	`offset = 0`
	`length = 240`
			DISPLAY TEXT Proactive command
			
			`Text String TLV =`
			`8D 81 F1 04 55 55…`

21	**Call the initDisplayText() method with a too long buffer**
	`qualifier = 0`
	`dcs = 4`
	`buffer = "XXXX…"`
	`offset = 0`
	`length = 241`
		HANDLER_OVERFLOW ToolkitException is thrown	

22	**Call the initDisplayText() without sending the command**		No proactive command shall be sent expected status is '9000'

6.2.7.3.4 Test Coverage

CRR number	Test case number
N1	8, 9, 10, 11, 12, 13, 14, 15, 16, 18, 19, 20
N2	15
N3	17
N4	22
N5	7
N6	16
P1	1
P2	2, 3, 4, 5, 6
C1	21

6.2.7.4 Method initGetInkey

Test Area Reference: API_2_PAH_INGKBB_BSS~~.~~

6.2.7.4.1 Conformance requirement

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

```
public void initGetInkey(byte qualifier,
                         byte dcs,
                         byte[] buffer,
                         short offset,
                         short length)
                  throws java.lang.NullPointerException,
                         java.lang.ArrayIndexOutOfBoundsException,
                         ToolkitException
```

6.2.7.4.1.1 Normal execution

CRRN1: The method shall build a GET INKEY proactive command in the ProactiveHandler, using qualifier, dcs and buffer parameters. Comprehension Required flags are set.

~~—~~CRRN2: A call to this method clears the handler then ~~initializes~~initialises it.

CRRN3: No TLV is selected after a call to the method.

CRRN4: The GET INKEY command is not sent by the method.

CRRN5: The Command Number may take any value between 01h and FEh.

CRRN6: If length is equal to zero, then the Text String TLV inserted in the command is a null text string TLV as defined in TS 11.14 [4].

6.2.7.4.1.2 Parameter errors

CRRP1: The method shall throw NullPointerException if buffer is null.

CRRP1: If offset or length or both would cause access outside array bounds, a ArrayIndexOutOfBoundsException shall be thrown.

6.2.7.4.1.3 Context errors

CRRC1: A ToolkitException.HANDLER_OVERFLOW shall be thrown if the ProactiveHandler is to small to put the requested data.

6.2.7.4.2 Test Suite files

Test Script: API_2_PAH_INGKBB_BSS_1.scr

Test Applet: ~~API_2_PAH_INGKBB_BSS_1.java~~

~~Load Script: API_2_PAH_INGKBB_BSS_1.ldr~~

~~Cleanup Script: API_2_PAH_INGKBB_BSS_1.clr~~

~~Parameter File:~~ API_2_PAH_INGKBB_BSS_1.java

Load Script: API_2_PAH_INGKBB_BSS_1.ldr

Cleanup Script: API_2_PAH_INGKBB_BSS_1.clr

Parameter File: API_2_PAH_INGKBB_BSS_1.par

6.2.7.4.3 Test procedure

Id	Description	API Expectation	APDU Expectation
1	**NULL as parameter to buffer** `buffer = NULL`	NullPointerException is thrown	

2	**offset > buffer.length**
	`buffer = "Text"`
	`offset = 5`
		ArrayIndexOutOfBoundsException is thrown	

3	**offset < 0**
	`buffer = "Text"`
	`offset = -1`
		ArrayIndexOutOfBoundsException is thrown	

4	**length > buffer.length**
	`buffer = "Text"`
	`offset = 0`
	`length = 5`
		ArrayIndexOutOfBoundsException is thrown	

5	**offset + length > buffer.length**
	`buffer = "Text"`
	`offset = 3`
	`length = 2`
		ArrayIndexOutOfBoundsException is thrown	

6	**length < 0**
	`buffer = "Text"`
	`offset = 3`
	`length = -1`
		ArrayIndexOutOfBoundsException is thrown	

7	**Successful call, buffer is the whole buffer**
	`qualifier = 0`
	`dcs = 4`
	`buffer = "TextA"`
	`offset = 0`
	`length = 5`
		No exception is thrown	

	Verify the command number value	Command number between 01h and FEh	

8	**Send the command**		GET INKEY Proactive command
			
			`qualifier = 00h`
			`dcs = 4`
			`Text = "TextA"`

9	**Successful call, buffer is part of a buffer with the end part**
	`qualifier = 0`
	`dcs = 4`
	`buffer = "12TextB"`
	`offset = 2`
	`length = 5`
			GET INKEY Proactive command
			
			`qualifier = 00h`
			`dcs = 4`
			`Text = "TextB"`

10	**Successful call, buffer is part of a buffer with the first part**
	`qualifier = 0`
	`dcs = 4`
	`buffer = "TextC12"`
	`offset = 0`
	`length = 5`
			GET INKEY ~~-~~Proactive command
			
			`qualifier = 00h`
			`dcs = 4`
			`Text = "TextC"`

11	**Successful call, buffer is part of a buffer**
	Send the command
	`qualifier = 0`
	`dcs = 4`
	`buffer = "12TextD34"`
	`offset = 2`
	`length = 5`
			GET INKEY ~~-~~Proactive command
			
			`qualifier = 00h`
			`dcs = 4`
			`Text = "TextD"`

12	**Successful call, qualifier = 81h** `qualifier = 81h`		GET INKEY Proactive command

	`dcs = 4`
	`buffer = "TextE"`
	`offset = 0`
	`length = 5`
			`qualifier = 81h`
			`dcs = 4`
			`Text = "TextE"`

13	**Successful call, DCS=0 (7 bits)**
	`qualifier = 0`
	`dcs = 0`
	`buffer = "TextF"`
	`offset = 0`
	`length = 5`
			GET INKEY Proactive command
			`qualifier = 00h`
			`dcs = 0`
			`Text = "TextF"`

14	**Successful call, DCS=8 (UCS2)**
	`qualifier = 0`
	`dcs = 8`
	`buffer = "TextG"`
	`offset = 0`
	`length = 5`
			GET INKEY Proactive command
			`qualifier = 00h`
			`dcs = 8`
			`Text = "TextG"`

15	**Call the initGetInkey() method with any value Then build and send a GET INKEY command**
	`qualifier = 0`
	`dcs = 4`
	`buffer = "TextHTextH"`
	`offset = 0`
	`length = 10`
			GET INKEY Proactive command
			`qualifier = 00h`
			`dcs = 4`
			`Text = "TextHTextH"`

~~16~~	~~**Successful call, text length is null Send the command**~~ ~~`qualifier = 0`~~ ~~`dcs = 4`~~ ~~`buffer = ""`~~ ~~`offset = 0`~~ ~~`length = 0`~~		~~GET INKEY Proactive command~~ ~~`qualifier = 00h`~~ ~~`Text String TLV = 8D 00`~~

16	**Successful call, text length is zero Send the command**
	`qualifier = 0`
	`dcs = 4`
	`buffer = "TextHTextH"`
	`offset = 0`
	`length = 0`
			GET INKEY Proactive command
			`qualifier = 00h`
			`Text String TLV = 8D 00`

17	**Select a TLV in the ProactiveHandler Call the initGetInkey() method Call the getValueLength() method**	UNAVAILABLE_ELEMENT ToolkitException is thrown by getValueLength()	

18	**Successful call, buffer length = 7Eh**
	`qualifier = 0`
	`dcs = 4`
	`buffer = "UUU…"`
	`offset = 0`
	`length = 7Eh`
			GET INKEY Proactive command
			`Text String TLV =`
			`8D 7F 04 55 55…`

19	**Successful call, buffer length = 7Fh**
	`qualifier = 0`
	`dcs = 4`
	`buffer = "UUU…"`
	`offset = 0`
	`length = 7Fh`
			GET INKEY Proactive command
			`Text String TLV = 8D 81 80 04 55 55…`

20	**Successful call, buffer length = 240**
	`Qualifier = 0`
	`dcs = 4`
	`buffer = "UUU…"`
	`offset = 0`
	`length = 240`
			GET INKEY Proactive command
			`Text String TLV =`
			`8D 81 F1 04 55 55…`

21	**Call the initGetInkey() method with a too long buffer**
	`qualifier = 0`
	`dcs = 4`
	`buffer = "XXXX…"`
	`offset = 0`
	`length = 241`
		HANDLER_OVERFLOW ToolkitException is thrown	

22	**Call the initGetInkey() without sending the command**		No proactive command shall be sent expected

			status is '9000'

6.2.7.4.4 Test Coverage

CRR number	Test case number
N1	8, 9, 10, 11, 12, 13, 14, 15, 16,18, 19, 20
N2	15
N3	17
N4	22
N5	7
N6	16
P1	1
P2	2, 3, 4, 5, 6
C1	21

6.2.7.5 Method initGetInput

Test Area Reference: API_2_PAH_INGPBB_BSSSS~~.~~

6.2.7.5.1 Conformance requirement

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

```
public void initGetInput(byte qualifier,
                         byte dcs,
                         byte[] buffer,
                         short offset,
                         short length,
                         short minRespLength,
                         short maxRespLength)
                  throws java.lang.NullPointerException,
                         java.lang.ArrayIndexOutOfBoundsException,
                         ToolkitException
```

6.2.7.5.1.1 Normal execution

CRRN1: The method shall build a GET INPUT proactive command in the ProactiveHandler, using qualifier, dcs, buffer, minRespLength and maxRespLength parameters. Comprehension Required flags are set.

~~—~~CRRN2: A call to this method clears the handler then ~~initializes~~initialises it.

CRRN3: No TLV is selected after a call to the method.

CRRN4: The GET INPUT command is not sent by the method.

CRRN5: The Command Number may take any value between 01h and FEh.

CRRN6: If length is equal to zero, then the Text String TLV inserted in the command is a null text string TLV as defined in TS 11.14 [4].

6.2.7.5.1.2 Parameter errors

CRRP1: The method shall throw NullPointerException if buffer is null.

CRRP2: If offset or length or both would cause access outside array bounds, a ArrayIndexOutOfBoundsException shall be thrown.

6.2.7.5.1.3 Context errors

CRRC1: A ToolkitException.HANDLER_OVERFLOW shall be thrown if the ProactiveHandler is to small to put the requested data.

6.2.7.5.2 Test Suite files

Test Script:	API_2_PAH_INGPBB_BSSSS_1.scr
Test Applet:	~~API_2_PAH_INGPBB_BSSSS_1.java~~
~~Load Script:~~	~~API_2_PAH_INGPBB_BSSSS_1.ldr~~
~~Cleanup Script:~~	~~API_2_PAH_INGPBB_BSSSS_1.clr~~
~~Parameter File:~~	API_2_PAH_INGPBB_BSSSS_1.java
Load Script:	API_2_PAH_INGPBB_BSSSS_1.ldr
Cleanup Script:	API_2_PAH_INGPBB_BSSSS_1.clr
Parameter File:	API_2_PAH_INGPBB_BSSSS_1.par

6.2.7.5.3 Test procedure

Id	Description	API Expectation	APDU Expectation
1	**NULL as parameter to buffer** `buffer = NULL`	NullPointerException is thrown	

2	**offset > buffer.length**
	`buffer = "Text"`
	`offset = 5`
		ArrayIndexOutOfBoundsException is thrown	

3	**offset < 0**
	`buffer = "Text"`
	`offset = -1`
		ArrayIndexOutOfBoundsException is thrown	

4	**length > buffer.length**
	`buffer = "Text"`
	`offset = 0`
	`length = 5`
		ArrayIndexOutOfBoundsException is thrown	

5	**offset + length > buffer.length**
	`buffer = "Text"`
	`offset = 3`
	`length = 2`
		ArrayIndexOutOfBoundsException is thrown	

6	**length < 0**
	`buffer = "Text"`
	`offset = 3`
	`length = -1`
		ArrayIndexOutOfBoundsException is thrown	

7	**Successful call, buffer is the whole buffer**
	
	`qualifier = 0`
	`dcs = 4`
	`buffer = "TextA"`
	`offset = 0`
	`length = 5`
	`minRespLength = 00h`
	`maxRespLength = FFh`
		No exception is thrown	

	~~**Verify the command number value**~~	~~Command number between 01h and FEh~~	
	Verify the command number value	Command number between 01h and FEh	

8	**Send the command**		GET INPUT Proactive command
			
			`qualifier = 00h`
			`dcs = 4`
			`Text = "TextA"`
			`Min Length = 00h`
			`Max Length = FFh`

9	**Successful call, buffer is part of a buffer with the end part**
	Send the command
	`qualifier = 0`
	`dcs = 4`
	`buffer = "12TextB"`
	`offset = 2`
	`length = 5`
	`minRespLength = 10h`
	`maxRespLength = FFh`
			GET INPUT Proactive command
			
			`qualifier = 00h`
			`dcs = 4`
			`Text = "TextB"`
			`Min Length = 10h`
			`Max Length = FFh`

10	**Successful call, buffer is part of a buffer with the first part**
	Send the command
	`qualifier = 0`
	`dcs = 4`
	`buffer = "TextC12"`
	`offset = 0`
	`length = 5`
	`minRespLength = FFh`
	`maxRespLength = FFh`
			GET INPUT Proactive command
			
			`qualifier = 00h`
			`dcs = 4`
			`Text = "TextC"`
			`Min Length = FFh`
			`Max Length = FFh`

11	**Successful call, buffer is part of a buffer**
	Send the command
	`qualifier = 0`
	`dcs = 4`
	`buffer = "12TextD34"`
	`offset = 2`
	`length = 5`
	`minRespLength = 00h`
	`maxRespLength = 00h`
			GET INPUT Proactive command
			
			`qualifier = 00h`
			`dcs = 4`
			`Text = "TextD"`
			`Min Length = 00h`
			`Max Length = 00h`

12	**Successful call, qualifier = 81h**
	`qualifier = 81h`
	`dcs = 4`
	`buffer = "TextE"`
	`offset = 0`
	`length = 5`
	`minRespLength = 00h`
	`maxRespLength = 10h`
			GET INPUT Proactive command
			
			`qualifier = 81h`
			`dcs = 4`
			`Text = "TextE"`
			`Min Length = 00h`
			`Max Length = 10h`

13	**Successful call, DCS=0 (7 bits)**
	`qualifier = 0`
	`dcs = 0`
	`buffer = "TextF"`
	`offset = 0`
	`length = 5`
	`minRespLength = 10h`
	`maxRespLength = 10h`
			GET INPUT Proactive command
			
			`qualifier = 00h`
			`dcs = 0`
			`Text = "TextF"`
			`Min Length = 10h`
			`Max Length = 10h`

14	**Successful call, DCS=8 (UCS2)**
	`qualifier = 0`
	`dcs = 8`
	`buffer = "TextG"`
	`offset = 0`
	`length = 5`
	`minRespLength = 00h`
	`maxRespLength = FFh`
			GET INPUT Proactive command
			
			`qualifier = 00h`
			`dcs = 8`
			`Text = "TextG"`
			`Min Length = 00h`
			`Max Length = FFh`

15	**Call the initGetInput() method with any value**
	Then build and send a GET INPUT command
	`qualifier = 0`
	`dcs = 4`
	`buffer = "TextHTextH"`
	`offset = 0`
	`length = 10`
	`minRespLength = 00h`
	`maxRespLength = 10h`
			GET INPUT Proactive command
			
			`qualifier = 00h`
			`dcs = 4`
			`Text = "TextHTextH"`
			`Min Length = 00h`
			`Max Length = 10h`

~~16~~	~~**Successful call, text length is null**~~ ~~**Send the command**~~ ~~qualifier = 0~~ ~~dcs = 4~~ ~~buffer = ""~~ ~~offset = 0~~ ~~length = 0~~ ~~minRespLength = 00h~~ ~~maxRespLength = 10h~~		~~GET INPUT Proactive command~~ ~~qualifier = 00h~~ ~~Text String TLV = 8D 00~~ ~~Min Length = 00h~~ ~~Max Length = 10h~~

16	**Successful call, text length is zero**
	Send the command
	`qualifier = 0`
	`dcs = 4`
	`buffer = "TextHTextH"`
	`offset = 0`
	`length = 0`
	`minRespLength = 00h`
	`maxRespLength = 10h`
			GET INPUT Proactive command
			
			`qualifier = 00h`
			`Text String TLV = 8D 00`
			`Min Length = 00h`
			`Max Length = 10h`

17	**Select a TLV in the ProactiveHandler** **Call the initGetInput() method**	UNAVAILABLE_ELEMENT ToolkitException is thrown by	

	Call the getValueLength() method	getValueLength()	

18	**Successful call, buffer length = 7Eh**
	
	`qualifier = 0`
	`dcs = 4`
	`buffer = "UUU…"`
	`offset = 0`
	`length = 7Eh`
	`minRespLength = 00h`
	`maxRespLength = 10h`
			GET INPUT Proactive command
			
			`Text String TLV =`
			`8D 7F 04 55 55…`
			`Min Length = 00h`
			`Max Length = 10h`

19	**Successful call, buffer length = 7Fh**
	
	`qualifier = 0`
	`dcs = 4`
	`buffer = "UUU…"`
	`offset = 0`
	`length = 7Fh`
	`minRespLength = 00h`
	`maxRespLength = 10h`
			GET INPUT Proactive command
			
			`Text String TLV = 8D 81`
			`80 04 55 55…`
			`Min Length = 00h`
			`Max Length = 10h`

20	**Successful call, buffer length = 236**
	
	`Qualifier = 0`
	`dcs = 4`
	`buffer = "UUU…"`
	`offset = 0`
	`length = 236`
	`minRespLength = 00h`
	`maxRespLength = 10h`
			GET INPUT Proactive command
			
			`Text String TLV =`
			`8D 81 ED 04 55 55…`

21	**Call the initGetInput() method with a too long buffer**
	`qualifier = 0`
	`dcs = 4`
	`buffer = "XXXX…"`
	`offset = 0`
	`length = 237`
	`minRespLength = 00h`
	`maxRespLength = 10h`
		HANDLER_OVERFLOW ToolkitException is thrown	

22	**Call the initGetInput() without sending the command**		No proactive command shall be sent expected status is '9000'

6.2.7.5.4 Test Coverage

CRR number	**Test case number**
N1	8, 9, 10, 11, 12, 13, 14, 15,16,18, 19, 20
N2	15
N3	17
N4	22
N5	7
N6	16
P1	1
P2	2, 3, 4, 5, 6
C1	21

6.2.7.6 Method send

Test Area Reference: API_2_PAH_SEND~~.~~

6.2.7.6.1 Conformance requirement

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

```
public byte send()
```

6.2.7.6.1.1 Normal execution

CRRN1: The send() method send the current proactive command to the mobile.

CRRN2: The returned byte is equal to general result of the command (first byte of Result TLV in Terminal Response).

CRRN3: The handler remains unchanged after a call to send() method until the use of initXX() or appendTLV().

CRRN4: There is no invocation of select() or deselect() method.

CRRN5: A pending toolkit applet transaction at the method invocation is aborted.

~~6.2.7.6.1.2 Parameter errors~~

~~No requirements.~~

6.2.7.6.1.3 Context errors

CRRC1: A ToolkitException.UNAVAILABLE_ELEMENT shall be thrown is the Result Simple TLV is missing in Terminal Response.

CRRC2: A ToolkitException.OUT_OF_TLV_BOUNDARIES shall be thrown if the general result byte is missing in the Result Simple TLV in Terminal Response.

CRRC3 : A ToolkitException COMMAND_NOT_ALLOWED shall be thrown if the proactive command to be sent is not allowed by the SIM Toolkit Framework.

CRRC4 : A ToolkitException COMMAND_NOT_ALLOWED shall be thrown if one parameter of the proactive command to be sent is not allowed by the SIM Toolkit Framework.

6.2.7.6.2 Test Suite files

Test Script:	API_2_PAH_SEND_1.scr
Test Applet:	~~API_2_PAH_SEND_1.java~~
~~Load Script:~~	~~API_2_PAH_SEND_1.ldr~~
~~Cleanup Script:~~	~~API_2_PAH_SEND_1.clr~~
~~Parameter File:~~	API_2_PAH_SEND_1.java
Load Script:	API_2_PAH_SEND_1.ldr
Cleanup Script:	API_2_PAH_SEND_1.clr
Parameter File:	API_2_PAH_SEND_1.par

6.2.7.6.3 Test procedure

Id	Description	API Expectation	APDU Expectation

1	**Build and send a DISPLAY TEXT command**
	`qualifier = 00h`
	`dcs = 04h`
	`buffer = 'Text'`
			DISPLAY TEXT Proactive command

2	**Terminal Response with General Result = 00** `Result TLV = 03 01 00 (command performed successfully)`	Result of send() is 00h	

3	**Build and send a DISPLAY TEXT command**
	`qualifier = 00h`
	`dcs = 04h`
	`buffer = 'Text'`
			DISPLAY TEXT Proactive command

4	**Terminal Response with General Result = 01, without Additional information on result** `Result TLV = 03 01 01 (command performed with partial comprehension)`	Result of send() is 01h	

5	**Build and send a DISPLAY TEXT command**
	`qualifier = 00h`
	`dcs = 04h`
	`buffer = 'Text'`
			DISPLAY TEXT Proactive command

6	**Terminal Response with General Result = 01, with Additional information on result** `Result TLV = 03 02 01 55 (command performed with partial comprehension)`	Result of send() is 01h	

7	**Build and send a DISPLAY TEXT command**
	`qualifier = 00h`
	`dcs = 04h`
	`buffer = 'Text'`
			DISPLAY TEXT Proactive command

8	**Terminal Response with General Result = 02** `Result TLV = 03 04 02 65 43 21 (Missing information)`	Result of send() is 02h	

9	**Build and send a 7Fh byte command (DISPLAY TEXT)**
	`qualifier = 00h`
	`dcs = 04h`
	`buffer = "UUUUU…"`
	`length = 73h`
			DISPLAY TEXT Proactive command
			
			`BER-TLV = D0 7F`
			`Text String TLV = 8D 74 04 55 55 55…`

10	**Build and send a 80h byte command (DISPLAY TEXT)**
	`qualifier = 00h`
	`dcs = 04h`
	`buffer = "UUUUU…"`
	`length = 74h`
			DISPLAY TEXT Proactive command
			
			`BER-TLV = D0 81 80`
			`Text String TLV = 8D 75 04 55 55 55…`

11	**Build and send a maximum length command (length of the handler should be 253)**
	
	`DISPLAY TEXT:`
	`Qualifier = 0`
	`dcs = 4`
	`buffer = "UUU…"`
	`offset = 0`
	`length = 240`
			DISPLAY TEXT Proactive command
			
			`BER-TLV = D0 81 FD`
			`Text String TLV = 8D 81 F1 04 55 55…`

12	**Verify that the Proactive Handler is not modified after a send()** **Build a DISPLAY TEXT command** **Copy ProactiveHandler to source byte array** **Send command** **Copy ProactiveHandler to destination byte array** **Compare source and destination**	Source and destination are identical	
13	**Build and send a DISPLAY TEXT command** **Verify there is no invocation of -select() or**		DISPLAY TEXT Proactive command

	deselect() method.		
14	**Build and send a DISPLAY TEXT command**		DISPLAY TEXT Proactive command

	Terminal Response with 2 Result TLV
	
	`1st Result TLV = 03 02 02 12`
	`2nd Result TLV = 03 03 03 34 56`
		Result of send() is 02h	

15	**Build and send a DISPLAY TEXT command**		DISPLAY TEXT Proactive command
	Terminal Response without Result Simple TLV	ToolkitException.UNAVAILABLE_ELEMENT is thrown by send()	
16	**Build and send a DISPLAY TEXT command**		DISPLAY TEXT Proactive command
	Terminal Response without general result byte in the Simple TLV **Result TLV = 03 00**	ToolkitException.OUT_OF_TLV_BOUNDARIES is thrown by send()	

6.2.7.6.4 Test Coverage

CRR number	Test case number
N1	1, 3, 5, 7, 9, 10, 11, 12, 13, 14
N2	2, 4, 6, 8, 14
N3	12
N4	13
N5	To be checked in Framework tests and insert here cross reference
C1	15
C2	16
C3	checked in the Framework test : FWK_PCS_PCCO (test case 1)
C4	checked in the Framework test : FWK_PCS_PCCO (test cases 2 to 3)

6.2.7.12 Method copyValue

Test Area Reference API_2_PAH_CPYVS_BSS~~.~~

6.2.7.12.1 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
```

6.2.7.12.1.1 Normal execution

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

CRRN2: returns ~~-~~dstOffset + dstLength.

6.2.7.12.1.2 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~~valueOffset is negative or valueOffset + dstLength > current TLV length, an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException OUT_OF_TLV_BOUNDARIES.

6.2.7.12.1.3 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.

6.2.7.12.2 Test Suite files

Test Script:	API_2_PAH_CPYVS_BSS_1.scr
Test Applet:	~~API_2_PAH_CPYVS_BSS_1.java~~
~~Load Script:~~	~~API_2_PAH_CPYVS_BSS_1.ldr~~
~~Cleanup Script:~~	~~API_2_PAH_CPYVS_BSS_1.clr~~
~~Parameter File:~~	API_2_PAH_CPYVS_BSS_1.java
Load Script:	API_2_PAH_CPYVS_BSS_1.ldr
Cleanup Script:	API_2_PAH_CPYVS_BSS_1.clr
Parameter File:	API_2_PAH_CPYVS_BSS_1.par

6.2.7.12.3 Test procedure

Id	Description	API Expectation	APDU Expectation
~~1~~	~~**Initialize the handler**~~ ~~**Select a TLV**~~		
1	**Initialise the handler** **Select a TLV**		
	`copyValue() with a null dstBuffer`	NullPointerException is thrown	
2	**initDisplayText() with length = 15** `Select Text String TLV`		

	dstOffset > dstBuffer.length
	`dstBuffer.length = 5`
	`dstOffset = 6`
	`dstLength = 0`
		ArrayIndexOutOfBoundsException is thrown	

3	**dstOffset < 0**
	`dstBuffer.length = 5`
	`dstOffset = -1`
	`dstLength = 1`
		ArrayIndexOutOfBoundsException is thrown	

4	**dstLength >dstBuffer.length**
	`dstBuffer.length = 5`
	`dstOffset = 0`
	`dstLength = 6`
		ArrayIndexOutOfBoundsException is thrown	

5	**dstOffset + dstLength >dstBuffer.length**
	`dstBuffer.length = 5`
	`dstOffset = 3`
	`dstLength = 3`
		ArrayIndexOutOfBoundsException is thrown	

6	**dstLength < 0**	ArrayIndexOutOfBoundsExceptio	

	dstBuffer.length = 5
	dstOffset = 0
	dstLength = -1
		n is thrown	

7	**initDisplayText() with length = 5** Select Text String TLV		

	valueOffset > Text String Length
	valueOffset = 7
	dstBuffer.length = 15
	dstOffset = 0
	dstLength = 0
		ToolkitException.OUT_OF_TLV_BOUNDARIES is thrown	

8	**[Select Text String TLV]**
	valueOffset < 0
	valueOffset = -1
	dstBuffer.length = 15
	dstOffset = 0
	dstLength = 1
		ToolkitException.OUT_OF_TLV_BOUNDARIES is thrown	

9	**[Select Text String TLV]**
	dstLength > Text String length
	valueOffset = 0
	dstBuffer.length = 15
	dstOffset = 0
	dstLength = 7
		ToolkitException.OUT_OF_TLV_BOUNDARIES is thrown	

10	**[Select Text String TLV]**
	valueOffset + dstLength > Text String length
	valueOffset = 2
	dstBuffer.length = 15
	dstOffset = 0
	dstLength = 5
		ToolkitException.OUT_OF_TLV_BOUNDARIES is thrown	

~~11~~	~~**Initialize the handler**~~		
11	**Initialise the handler**		
	copyValue()	ToolkitException.UNAVAILABLE_ELEMENT is thrown	

12	**initDisplayText()**
	dcs = 4
	buffer = 00 01 … 0F
	Select Text String TLV		

	Successful call
	valueOffset = 0
	dstBuffer.length = 17
	dstOffset = 0
	dstLength = 17
		Result of copyValue() is 17	

13	**Compare buffer** buffer = 04 00 01 … 0F	Result is 00h	
~~14~~	~~**initialize dstBuffer**~~ ~~dstBuffer = 55 55 … 55~~		
14	**initialise dstBuffer** dstBuffer = 55 55 … 55		

	Successful call
	valueOffset = 2
	dstBuffer.length = 20
	dstOffset = 3
	dstLength = 12
		Result of copyValue() is 15	

15	**Compare buffer** buffer = 55 55 55 01 02 03 04 05 06 07 08 09 0A 0B 0C 55 55 55 55 55	Result is 00h	

6.2.7.12.4 Test Coverage

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

6.2.7.13 Method compareValue

Test Area Reference API_2_PAH_CPRVS_BSS~~.~~

6.2.7.13.1 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
```

6.2.7.13.1.1 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 simple TLV List is less than that in compareBuffer.

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

6.2.7.13.1.2 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~~valueOffset is negative or valueOffset + dstLength > current TLV length, an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException OUT_OF_TLV_BOUNDARIES.

6.2.7.13.1.3 Context errors

CRRC1: if the handler is busy an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException HANDLER_NOT_AVAILABLE.

CRRC2: in case of unavailable TLV element an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException UNAVAILABLE_ELEMENT.

6.2.7.13.2 Test Suite files

Test Script:	API_2_PAH_CPRVS_BSS_1.scr
Test Applet:	~~API_2_PAH_CPRVS_BSS_1.java~~
~~Load Script:~~	~~API_2_PAH_CPRVS_BSS_1.ldr~~

~~Cleanup Script: API_2_PAH_CPRVS_BSS_1.clr~~

~~Parameter File:~~ API_2_PAH_CPRVS_BSS_1.java

Load Script: API_2_PAH_CPRVS_BSS_1.ldr

Cleanup Script: API_2_PAH_CPRVS_BSS_1.clr

Parameter File: API_2_PAH_CPRVS_BSS_1.par

6.2.7.13.3 Test procedure

Id	Description	API Expectation	APDU Expectation
~~1~~	~~Initialize the handler~~ ~~Select a TLV~~		
1	Initialise the handler Select a TLV		
	compareValue() with a null compareBuffer	NullPointerException is thrown	
2	**initDisplayText() with length = 15** `Select Text String TLV`		

	compareOffset > compareBuffer.length
	`compareBuffer.length = 5`
	`compareOffset = 6`
	`compareLength = 0`
		ArrayIndexOutOfBoundsException is thrown	

3	**compareOffset < 0**
	`compareBuffer.length = 5`
	`compareOffset = -1`
	`compareLength = 1`
		ArrayIndexOutOfBoundsException is thrown	

4	**compareLength >compareBuffer.length**
	`compareBuffer.length = 5`
	`compareOffset = 0`
	`compareLength = 6`
		ArrayIndexOutOfBoundsException is thrown	

5	**compareOffset + compareLength >compareBuffer.length**
	`compareBuffer.length = 5`
	`compareOffset = 3`
	`compareLength = 3`
		ArrayIndexOutOfBoundsException is thrown	

6	**compareLength < 0**
	compareBuffer.length = 5
	compareOffset = 0
	compareLength = -1
		ArrayIndexOutOfBoundsException is thrown	

7	**initDisplayText() with length = 5** **Select Text String TLV**		

	valueOffset > Text String Length
	`valueOffset = 7`
	`compareBuffer.length = 15`
	`compareOffset = 0`
	`compareLength = 0`
		ToolkitException.OUT_OF_TLV_BOUNDARIES is thrown	

8	**[Select Text String TLV]**
	`valueOffset < 0`
	`valueOffset = -1`
	`compareBuffer.length = 15`
	`compareOffset = 0`
	`compareLength = 1`
		ToolkitException.OUT_OF_TLV_BOUNDARIES is thrown	

9	**[Select Text String TLV]**
	`compareLength > Text String length`
	`valueOffset = 0`
	`compareBuffer.length = 15`
	`compareOffset = 0`
	`compareLength = 7`
		ToolkitException.OUT_OF_TLV_BOUNDARIES is thrown	

10	**[Select Text String TLV]**
	`valueOffset + compareLength > Text String length`
	`valueOffset = 2`
	`compareBuffer.length = 15`
	`compareOffset = 0`
	`compareLength = 5`
		ToolkitException.OUT_OF_TLV_BOUNDARIES is thrown	

~~11~~	~~Initialize the handler~~		
11	Initialise the handler		

	~~compareValue()~~	~~ToolkitException.UNAVAILABLE_ ELEMENT is thrown~~	
	compareValue()	ToolkitException.UNAVAILABLE _ELEMENT is thrown	

12	**initDisplayText()**
	`dcs = 4`
	`buffer = 00 01 … 0F`
	`Select Text String TLV`		

	~~Initialize compareBuffer~~ ~~`compareBuffer =`~~ ~~`04 00 01 … 0F`~~		

	Initialise compareBuffer
	`compareBuffer =`
	`04 00 01 … 0F`		

	Compare buffers
	`valueOffset = 0`
	`compareOffset = 0`
	`compareLength = 17`
		Result is 00h	

~~13~~	**~~Initialize compareBuffer~~** ~~`compareBuffer =`~~ ~~`04 00 01 02 03`~~ ~~`04 05 06 07 08`~~ ~~`05 0A 0B 0C 0D`~~ ~~`0E 10`~~		

13	**Initialise compareBuffer**
	`compareBuffer =`
	`04 00 01 02 03`
	`04 05 06 07 08`
	`05 0A 0B 0C 0D`
	`0E 10`		

	Compare buffers with same parameters	Result is -1	
~~14~~	**~~Initialize compareBuffer~~** ~~`compareBuffer =`~~ ~~`03 00 01 … 0F`~~		

14	**Initialise compareBuffer**
	`compareBuffer =`
	`03 00 01 … 0F`		

	Compare buffers with same parameters	Result is +1	
~~15~~	**~~Initialize compareBuffer~~** ~~`compareBuffer =`~~ ~~`55 55 55 01 02`~~ ~~`03 04 05 06 07`~~ ~~`08 09 0A 0B 0C`~~ ~~`55 55 55 55 55`~~		

15	**Initialise compareBuffer**
	`compareBuffer =`
	`55 55 55 01 02`
	`03 04 05 06 07`
	`08 09 0A 0B 0C`
	`55 55 55 55 55`		

	Compare buffers
	`valueOffset = 2`
	`compareOffset = 3`
	`compareLength = 12`
		Result is 00h	

~~16~~	**~~Initialize compareBuffer~~** ~~`compareBuffer =`~~ ~~`55 55 55 02 01`~~ ~~`03 04 05 06 07`~~ ~~`08 09 0A 0B 0C`~~ ~~`55 55 55 55 55`~~		

16	**Initialise compareBuffer**
	`compareBuffer =`
	`55 55 55 02 01`
	`03 04 05 06 07`
	`08 09 0A 0B 0C`
	`55 55 55 55 55`		

	Compare buffers with same parameters	Result is -1	
~~17~~	**~~Initialize compareBuffer~~** ~~`compareBuffer =`~~		

	~~55 55 55 01 02~~ ~~03 04 05 06 07~~ ~~08 09 0A 0A 0D~~ ~~55 55 55 55 55~~		

17	**Initialise compareBuffer**
	`compareBuffer =`
	`55 55 55 01 02`
	`03 04 05 06 07`
	`08 09 0A 0A 0D`
	`55 55 55 55 55`		

	Compare buffers with same parameters	Result is +1	
~~18~~	~~**Initialize compareBuffer**~~ ~~compareBuffer =~~ ~~55 55 55 99 03~~ ~~03 04 05 06 07~~ ~~08 09 0A 0B 0C~~ ~~55 55 55 55 55~~		

18	**Initialise compareBuffer**
	`compareBuffer =`
	`55 55 55 99 03`
	`03 04 05 06 07`
	`08 09 0A 0B 0C`
	`55 55 55 55 55`		

	Compare buffers with same parameters	Result is +1	

6.2.7.13.4 Test Coverage

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

6.2.7.15 Method findAndCopyValue(byte tag, byte occurence, short valueOffset, byte[] dstBuffer, short dstOffset, short dstLength)

Test Area Reference API_2_PAH_FACYBBS_BSS~~.~~

6.2.7.15.1 Conformance requirement

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

```
public short findAndCopyValue(byte tag,
                              byte occurence,
                              short valueOffset,
                              byte[] dstBuffer,
                              short dstOffset,
                              short dstLength)
                       throws java.lang.NullPointerException,
                              java.lang.ArrayIndexOutOfBoundsException,
                              ToolkitException
```

6.2.7.15.1.1 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.

6.2.7.15.1.2 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~~valueOffset is negative or valueOffset + dstLength > current TLV length, an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException OUT_OF_TLV_BOUNDARIES.

6.2.7.15.1.3 Context errors

CRRC1: if the handler is busy an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException HANDLER_NOT_AVAILABLE.

6.2.7.15.2 Test Suite files

Test Script:	API_2_PAH_FACYBBS_BSS_1.scr
Test Applet:	~~API_2_PAH_FACYBBS_BSS_1.java~~
~~Load Script:~~	~~API_2_PAH_FACYBBS_BSS_1.ldr~~
~~Cleanup Script:~~	~~API_2_PAH_FACYBBS_BSS_1.clr~~
~~Parameter File:~~	API_2_PAH_FACYBBS_BSS_1.java
Load Script:	API_2_PAH_FACYBBS_BSS_1.ldr
Cleanup Script:	API_2_PAH_FACYBBS_BSS_1.clr
Parameter File:	API_2_PAH_FACYBBS_BSS_1.par

6.2.7.15.3 Test procedure

Id	Description	API Expectation	APDU Expectation
~~1~~	~~**Initialize the handler**~~		
1	**Initialise the handler** **findAndCopyValue() with a null dstBuffer**	NullPointerException is thrown	

2	**initDisplayText() with length = 15**
	dstOffset > dstBuffer.length
	`tag = 0Dh, occurrence = 1`
	`valueOffset = 0`
	`dstBuffer.length = 5`
	`dstOffset = 6`
	`dstLength = 0`
		ArrayIndexOutOfBoundsException is thrown	

3	**dstOffset < 0**
	`dstBuffer.length = 5`
	`dstOffset = -1`
	`dstLength = 1`
		ArrayIndexOutOfBoundsException is thrown	

4	**dstLength >dstBuffer.length**
	`dstBuffer.length = 5`
	`dstOffset = 0`
	`dstLength = 6`
		ArrayIndexOutOfBoundsException is thrown	

5	**dstOffset + dstLength >dstBuffer.length**
	`dstBuffer.length = 5`
	`dstOffset = 3`
	`dstLength = 3`
		ArrayIndexOutOfBoundsException is thrown	

6	**dstLength < 0** `dstBuffer.length = 5`	ArrayIndexOutOfBoundsException is thrown	

	`dstOffset = 0`
	`dstLength = -1`		

7	**initDisplayText() with length = 5**		

	valueOffset > Text String Length
	`tag = 0Dh, occurrence = 1`
	`valueOffset = 7`
	`dstBuffer.length = 15`
	`dstOffset = 0`
	`dstLength = 0`
		ToolkitException.OUT_OF_TLV_ BOUNDARIES is thrown	

8	**valueOffset < 0**
	`valueOffset = -1`
	`dstBuffer.length = 15`
	`dstOffset = 0`
	`dstLength = 1`
		ToolkitException.OUT_OF_TLV_ BOUNDARIES is thrown	

9	**dstLength > Text String length**
	`valueOffset = 0`
	`dstBuffer.length = 15`
	`dstOffset = 0`
	`dstLength = 7`
		ToolkitException.OUT_OF_TLV_ BOUNDARIES is thrown	

10	**valueOffset + dstLength > Text String length**
	`valueOffset = 2`
	`dstBuffer.length = 15`
	`dstOffset = 0`
	`dstLength = 5`
		ToolkitException.OUT_OF_TLV_ BOUNDARIES is thrown	

11	**InitDisplayText()**		
	Select a TLV (tag 02h)		

	findAndCopyValue()
	`tag = 0Dh`
	`occurrence = 2`
		ToolkitException.UNAVAILABLE _ELEMENT is thrown	

	Call the getValueLength() method	ToolkitException.UNAVAILABLE _ELEMENT is thrown.	

12	**initDisplayText()**
	`dcs = 4`
	`buffer = 00 01 … 0F`		

	Successful call
	`tag = 0Dh, occurrence = 1`
	`valueOffset = 0`
	`dstBuffer.length = 17`
	`dstOffset = 0`
	`dstLength = 17`
		Result of findAndCopyValue() is 17	

13	**Compare buffer** `buffer = 04 00 01 … 0F`	Result is 00h	
~~14~~	~~**initialize dstBuffer**~~ ~~`dstBuffer = 55 55 … 55`~~		
14	**initialise dstBuffer** `dstBuffer = 55 55 … 55`		

	Successful call
	`tag = 0Dh, occurrence = 1`
	`valueOffset = 2`
	`dstBuffer.length = 20`
	`dstOffset = 3`
	`dstLength = 12`
		Result of findAndcopyValue() is 15	

15	**Compare buffer**
	`buffer =`
	`55 55 55 01 02`
	`03 04 05 06 07`
	`08 09 0A 0B 0C`
	`55 55 55 55 55`
		Result is 00h	

16	**Append a Text String TLV**
	`tag = 0D`
	`buffer = 00 11 22 33 44 55 (no specific DCS byte)`		

	Successful call
	`tag = 0Dh, occurrence = 1`
	`valueOffset = 0`
	`dstBuffer.length = 17`
	`dstOffset = 0`
	`dstLength = 17`
		Result of findAndCopyValue() is 17	

17	**Compare buffer** `buffer = 04 00 01 … 0F`	Result is 00h	

18	**Successful call**
	`tag = 0Dh, occurrence = 2`
	`valueOffset = 0`
	`dstBuffer.length = 6`
	`dstOffset = 0`
	`dstLength = 6`
		Result of findAndCopyValue() is 6	

19	**Compare buffer** `buffer = 00 11 22 33 44 55`	Result is 00h	

20	**initDisplayText()**
	`dcs = 4`
	`buffer = 00 01 … 0F`		

	Successful call (with tag 8Dh)
	`tag = 8Dh`
	`occurrence = 1`
	`valueOffset = 0`
	`dstBuffer.length = 17`
	`dstOffset = 0`
	`dstLength = 17`
		Result of findAndcopyValue() is 17	

21	**Compare buffer** `buffer = 04 00 01 … 0F`	Result is 00h	
22	**Append tag 0Fh** `buffer = 00 01 … 0F`		

	Successful call (with tag 8Fh)
	`tag = 8Fh`
	`occurrence = 1`
	`valueOffset = 0`
	`dstBuffer.length = 16`
	`dstOffset = 0`
	`dstLength = 16`
		Result of findAndcopyValue() is 16	

23	**Compare buffer** `buffer = 00 01 … 0F`	Result is 00h	

6.2.7.15.4 Test Coverage

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

6.2.7.17 Method findAndCompareValue(byte tag, byte occurence, short valueOffset, byte[] compareBuffer, short compareOffset, short compareLength)

Test Area Reference API_2_PAH_FACRBBS_BSS~~.~~

6.2.7.17.1 Conformance requirement

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

```
public byte findAndCompareValue(byte tag,
                                byte occurence,
                                short valueOffset,
                                byte[] compareBuffer,
                                short compareOffset,
                                short compareLength)
                       throws   java.lang.NullPointerException,
                                java.lang.ArrayIndexOutOfBoundsException,
                                ToolkitException
```

6.2.7.17.1.1 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 simple TLV is less than that in compareBuffer -1 is returned.

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

CRRN6: The search method is comprehension required flag independent.

6.2.7.17.1.2 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~~valueOffset is negative or valueOffset + dstLength > current TLV length, 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.

6.2.7.17.1.3 Context errors

CRRC1: if the handler is busy an instance of ToolkitException shall be thrown. The reason code shall be ToolkitException HANDLER_NOT_AVAILABLE.

6.2.7.17.2 Test Suite files

Test Script:	API_2_PAH_FACRBBS_BSS_1.scr
Test Applet:	~~API_2_PAH_FACRBBS_BSS_1.java~~
~~Load Script:~~	~~API_2_PAH_FACRBBS_BSS_1.ldr~~
~~Cleanup Script:~~	~~API_2_PAH_FACRBBS_BSS_1.clr~~
~~Parameter File:~~	API_2_PAH_FACRBBS_BSS_1.java
Load Script:	API_2_PAH_FACRBBS_BSS_1.ldr
Cleanup Script:	API_2_PAH_FACRBBS_BSS_1.clr
Parameter File:	API_2_PAH_FACRBBS_BSS_1.par

6.2.7.17.3 Test procedure

Id	Description	API Expectation	APDU Expectation
~~1~~	~~**Initialize the handler**~~		
1	**Initialise the handler**		
	findAndCompareValue() with a null compareBuffer	NullPointerException is thrown	
2	**initDisplayText() with length = 15**		

	compareOffset > compareBuffer.length
	`tag = 0Dh, occurrence = 1`
	`valueOffset = 0`
	`compareBuffer.length = 5`
	`compareOffset = 6`
	`compareLength = 0`
		ArrayIndexOutOfBoundsException is thrown	

3	**compareOffset < 0**
	`compareBuffer.length = 5`
	`compareOffset = -1`
	`compareLength = 1`
		ArrayIndexOutOfBoundsException is thrown	

4	**compareLength >compareBuffer.length**
	`compareBuffer.length = 5`
	`compareOffset = 0`
	`compareLength = 6`
		ArrayIndexOutOfBoundsException is thrown	

5	**compareOffset + compareLength** `>compareBuffer.length`
	`compareBuffer.length = 5`
	`compareOffset = 3`
	`compareLength = 3`
		ArrayIndexOutOfBoundsException is thrown	

6	**compareLength < 0**
	`compareBuffer.length = 5`
	`compareOffset = 0`
	`compareLength = -1`
		ArrayIndexOutOfBoundsException is thrown	

7	**initDisplayText() with length = 5**		

	valueOffset > Text String Length
	`tag = 0Dh, occurrence = 1`
	`valueOffset = 7`
	`compareBuffer.length = 15`
	`compareOffset = 0`
	`compareLength = 0`
		ToolkitException.OUT_OF_TLV_BOUNDARIES is thrown	

8	**valueOffset < 0**
	`valueOffset = -1`
	`compareBuffer.length = 15`
	`compareOffset = 0`
	`compareLength = 1`
		ToolkitException.OUT_OF_TLV_BOUNDARIES is thrown	

9	**compareLength > Text String length**
	`valueOffset = 0`
	`compareBuffer.length = 15`
	`compareOffset = 0`
	`compareLength = 7`
		ToolkitException.OUT_OF_TLV_BOUNDARIES is thrown	

10	**valueOffset + compareLength > Text String length**
	`valueOffset = 2`
	`compareBuffer.length = 15`
	`compareOffset = 0`
	`compareLength = 5`
		ToolkitException.OUT_OF_TLV_BOUNDARIES is thrown	

11	**Invalid parameter** `occurrence = 0`	ToolkitException.BAD_INPUT_PARAMETER is thrown	
12	**InitDisplayText()**		
	Select a TLV (tag 02h)		

	findAndCompareValue()
	`tag = 0Dh`
	`occurrence = 2`
		ToolkitException.UNAVAILABLE_ELEMENT is thrown	

	Call the getValueLength() method	ToolkitException.UNAVAILABLE_ELEMENT is thrown.	

13	**initDisplayText()**
	`dcs = 4`
	`buffer = 00 01 … 0F`		

	~~**Initialize compareBuffer**~~ ~~`compareBuffer =`~~ ~~`04 00 01 … 0F`~~		

	Initialise compareBuffer
	`compareBuffer =`
	`04 00 01 … 0F`		

	findAndCompareValue()
	`tag = 0Dh, occurrence = 1`
	`valueOffset = 0`
	`compareOffset = 0`
	`compareLength = 17`
		Result is 00h	

14	**Verify current TLV** `getValueLength()`	Result is 17	
~~15~~	~~**Initialize compareBuffer**~~ ~~`compareBuffer =`~~ ~~`04 00 01 … 10`~~		

15	**Initialise compareBuffer**
	`compareBuffer =`
	`04 00 01 … 10`		

	Compare buffers with same parameters	Result is -1	
~~16~~	~~**Initialize compareBuffer**~~ ~~`compareBuffer =`~~ ~~`03 00 01 … 0F`~~		

16	**Initialise compareBuffer**
	`compareBuffer =`
	`03 00 01 … 0F`		

	Compare buffers with same parameters	Result is +1	
~~17~~	~~**Initialize compareBuffer**~~ ~~`compareBuffer =`~~ ~~`55 55 55 01 02`~~ ~~`03 04 05 06 07`~~ ~~`08 09 0A 0B 0C`~~ ~~`55 55 55 55 55`~~		

17	**Initialise compareBuffer**
	`compareBuffer =`
	`55 55 55 01 02`
	`03 04 05 06 07`
	`08 09 0A 0B 0C`
	`55 55 55 55 55`		

	Compare buffers
	`valueOffset = 2`
	`compareOffset = 3`
	`compareLength = 12`
		Result is 00h	

~~18~~	~~**Initialize compareBuffer**~~ ~~`compareBuffer =`~~ ~~`55 55 55 02 01`~~ ~~`03 04 05 06 07`~~ ~~`08 09 0A 0B 0C`~~ ~~`55 55 55 55 55`~~		

18	**Initialise compareBuffer**
	`compareBuffer =`
	`55 55 55 02 01`
	`03 04 05 06 07`
	`08 09 0A 0B 0C`
	`55 55 55 55 55`		

	Compare buffers with same parameters	Result is -1	
~~19~~	~~**Initialize compareBuffer**~~ ~~`compareBuffer =`~~ ~~`55 55 55 01 02`~~ ~~`03 04 05 06 07`~~ ~~`08 09 0A 0A 0D`~~ ~~`55 55 55 55 55`~~		

19	**Initialise compareBuffer**
	`compareBuffer =`
	`55 55 55 01 02`
	`03 04 05 06 07`
	`08 09 0A 0A 0D`
	`55 55 55 55 55`		

	Compare buffers with same parameters	Result is +1	
20	**append a Text String TLV**		

	`tag = 0Dh`
	`buffer = 00 11 22 33 44 55`		

	~~**Initialize compareBuffer**~~ ~~`compareBuffer =`~~ ~~`04 00 01 … 0F`~~		

	Initialise compareBuffer
	`compareBuffer =`
	`04 00 01 … 0F`		

	findAndCompareValue()
	`tag = 0Dh, occurrence = 1`
	`valueOffset = 0`
	`compareOffset = 0`
	`compareLength = 17`
		Result is 00h	

~~21~~	~~**Initialize compareBuffer**~~ ~~`compareBuffer =`~~ ~~`00 11 22 33 44 55`~~		

21	**Initialise compareBuffer**
	`compareBuffer =`
	`00 11 22 33 44 55`		

	findAndCompareValue()
	`tag = 0Dh, occurrence = 2`
	`valueOffset = 0`
	`compareOffset = 0`
	`compareLength = 6`
		Result is 00h	

~~22~~	~~**Initialize compareBuffer**~~ ~~`compareBuffer =`~~ ~~`00 11 22 33 44 66`~~		

22	**Initialise compareBuffer**
	`compareBuffer =`
	`00 11 22 33 44 66`		

	findAndCompareValue()
	`tag = 0Dh, occurrence = 2`
	`valueOffset = 0`
	`compareOffset = 0`
	`compareLength = 6`
		Result is –1	

23	**initDisplayText()**
	`dcs = 4`
	`buffer = 00 01 … 0F`		

	~~**Initialize compareBuffer**~~ ~~`CompareBuffer = 04 00 01 … 0F`~~		
	Initialise compareBuffer `CompareBuffer = 04 00 01 … 0F`		

	Successful call (with tag 8Dh)
	`tag = 8Dh, occurrence = 1`
	`valueOffset = 0`
	`compareBuffer.length = 17`
	`compareOffset = 0`
	`compareLength = 17`
		Result is 00h	

24	**Append tag 0Fh** `buffer = 00 01 … 0F`		
	~~**Initialize compareBuffer**~~ ~~`compareBuffer = 00 01 … 0F`~~		
	Initialise compareBuffer `compareBuffer = 00 01 … 0F`		

	Successful call (with tag 8Fh)
	`tag = 8Fh, occurrence = 1`
	`valueOffset = 0`
	`compareBuffer.length = 16`
	`compareOffset = 0`
	`compareLength = 16`
		Result is 00h	

~~25~~	~~**Initialize compareBuffer**~~ ~~`compareBuffer =0099 02 … 0F`~~		
25	**Initialise compareBuffer** `compareBuffer =0099 02 … 0F`		

	findAndCompareValue()
	`tag = 0Dh, occurrence = 1`
	`valueOffset = 0`
	`compareOffset = 0`
	`compareLength = 17`
		Result is +1	

6.2.7.17.4 Test Coverage

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

6.2.7.24 Method getCapacity

Test Area Reference: API_2_PAH_GCAP

6.2.7.24.1 Conformance Requirement

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

```
public byte getCapacity()
```

6.2.7.24.1.1 Normal execution

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

6.2.7.24.2 Test suite files

Test Script:	API_2_PAH_GCAP_1.scr
Test Applet:	API_2_PAH_GCAP_1.java
Load Script:	API_2_PAH_GCAP_1.ldr
Cleanup Script:	API_2_PAH_GCAP_1.clr
Parameter File:	API_2_PAH_GCAP_1.par

6.2.7.24.3 Test Procedure

Id	Description	API Expectation	APDU Expectation
1	**ProactiveHandler available** 1- Send envelope SMS-PP Formatted 2- The applet calls getTheHandler() 3- The applet calls getCapacity() on the ProactiveHandler 4- The applet fills the handler with the maximum capacity, using appendTLV() method 5- The applet calls clear() on the proactive handler 6- The applet fills the handler with the maximum capacity plus one, using appendTLV() method	1- Applet is triggered 2- No exception is thrown 3- No exception is thrown, the capacity shall not be null 4- No exception is thrown 5- No exception is thrown 6- HANDLER_OVERFLOW exception is thrown	

6.2.7.24.4 Test Coverage

CRR number	Test case number
N1	1

6.2.7.25 Method initCloseChannel

Test Area Reference: API_2_PAH_ICCHB

6.2.5.21.1 Conformance requirement

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

```
public void initCloseChannel(byte bChannelIdentifier)
```

6.2.7.25.1.1 Normal execution

CRRN1: The method shall build a Close Channel Proactive command, using Channel Identifier. Comprehension Required flags are set.

CRRN2: A call to this method clears the handler then initialises it with Close Channel Proactive command.

CRRN3: After the method invocation, no TLV is selected.

CRRN4: The CLOSE CHANNEL Proactive command is not sent by the method.

6.2.7.25.2 Test suite files

Test Script:	API_2_PAH_ICCHB_1.scr
Test Applet:	API_2_PAH_ICCHB_1.java
Load Script:	API_2_PAH_ICCHB_1.ldr
Cleanup Script:	API_2_PAH_ICCHB_1.clr
Parameter File:	API_2_PAH_ICCHB_1.par

6.2.7.25.3 Test procedure

Id	Description	API Expectation	APDU Expectation
0	`Applet1 is installed with maximum number of channel = 01.`		

1	**Call initCloseChannel() method**
	
	`1- Call ProactiveHandler.init() method to Open a Channel.`
	`Call the ProactiveHandler.send() method.`
		2- Applet1 is triggered.	1- OPEN CHANNEL proactive command is fetched.
			
			TERMINAL RESPONSE of

	`2- Send an EVENT_EVENT_DOWNLOAD_CHANNEL_STATUS Envelope.`
	
	`3- Call the ProactiveHandler.initCloseChannel() method with Channel Id = 01.`
	
	`4- Call the ProactiveHandler.send() method.`
	
	`5- Send an EVENT_EVENT_DOWNLOAD_CHANNEL_STATUS Envelope.`
		5- Applet1 is not triggered.	OPEN CHANNEL is sent to the SIM with Channel Id = 01.
			
			4- CLOSE CHANNEL proactive command is fetched.
			
			TERMINAL RESPONSE of CLOSE CHANNEL is sent to the SIM.

2	**Call the initCloseChannel () method with any value then build and send a CLOSE CHANNEL command**
	
	`1- Call ProactiveHandler.init() to Open a Channel and ProactiveHandler.send() methods.`
	
	`2- ProactiveHandler.initCloseChannel() with Channel Id = 2`
	
	`3- ProactiveHandler.initCloseChannel() with the Channel Id = 1.`
	
	`4- call the send() method.`
	
	`5- Send an EVENT_EVENT_DOWNLOAD_CHANNEL_STATUS Envelope.`
		5- Applet1 is not triggered.	1- OPEN CHANNEL proactive command is fetched.
			
			TERMINAL RESPONSE of OPEN CHANNEL is sent to the SIM with Channel Id = 01.
			
			4- CLOSE CHANNEL proactive command is fetched.
			
			TERMINAL RESPONSE of CLOSE CHANNEL is sent to the SIM.

3	**Select a TLV in the ProactiveHandler Call the initCloseChannel () method**
	
	`1- Call ProactiveHandler.init() method to open a Channel and call the ProactiveHandler.send() method. Select 1st TLV of the Proactive Handler.`
	
	`2- Call ProactiveHandler.initCloseChannel() method with Channel Id = 01.`
	
	`3- Call the ViewHandler.getValueLength() method.`
	
	`4- Call ProactiveHandler.send() method.`
		3- UNAVAILABLE_ELEMENT ToolkitException is thrown by getValueLength() method.	1- OPEN CHANNEL proactive command is fetched.
			
			TERMINAL RESPONSE of OPEN CHANNEL is sent to the SIM with Channel Id = 01.
			
			4- CLOSE CHANNEL proactive command is fetched.
			
			TERMINAL RESPONSE of CLOSE CHANNEL is sent to the SIM.

4	**Call the initCloseChannel() without sending the command**
	
	`1- Call ProactiveHandler.init() method to open a Channel and call the ProactiveHandler.send() method.`
	
	`2- Call the ProactiveHandler.initCloseChannel() method with Channel Id = 01 without ProactiveHandler.send().`
	
	`3- Send an EVENT_EVENT_DOWNLOAD_CHANNEL_STATUS Envelope.`
		3- Applet1 is triggered.	1- OPEN CHANNEL proactive command is fetched.
			
			TERMINAL RESPONSE of OPEN CHANNEL is sent to the SIM with Channel Id = 01.
			
			No proactive command shall be sent. Expected status is '9000'

6.2.5.21.1 Test Coverage

CRR number	Test case number
N1	1
N2	2
N3	3
N4	2, 4

6.2.8 Class ProactiveResponseHandler

6.2.8.20 Method getCapacity

Test Area Reference: API_2_PRH_GCAP

6.2.8.20.1 Conformance Requirement

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

```
public byte getCapacity()
```

6.2.8.20.1.1 Normal execution

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

6.2.8.20.2 Test suite files

Test Script:	API_2_PRH_GCAP_1.scr
Test Applet:	API_2_PRH_GCAP_1.java
Load Script:	API_2_PRH_GCAP_1.ldr
Cleanup Script:	API_2_PRH_GCAP_1.clr
Parameter File:	API_2_PRH_GCAP_1.par

6.2.8.20.3 Test Procedure

Id	Description	API Expectation	APDU Expectation
1	**ProactiveResponseHandler available** 1- Send envelope SMS-PP Formatted 2- The applet sends a proactive command 3- Fetch the proactive command and send Terminal Response 4- The applet calls method getCapacity() method 5- The applet calls method getLength() method	1- Applet is triggered 4-No exception is thrown 5- The Capacity result is greater or equal to getLength() result	2- 91 XX 3- The proactive command is fetched

6.2.8.20.4 Test Coverage

CRR number	Test case number
N1	1

6.2.8.21 Method getChannelIdentifier

Test Area Reference: API_2_PRH_GCID

6.2.8.21.1 Conformance Requirement:

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

```
public byte getChannelIdentifier()
                                        throws ToolkitException
```

6.2.8.21.1.1 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.

6.2.8.21.1.2 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 Simple TLV Channel Status length is equal to 0.

6.2.8.21.2 Test suite files

Test Script:	API_2_PRH_GCID_1.scr
Test Applet:	API_2_PRH_GCID_1.java
Load Script:	API_2_PRH_GCID_1.ldr
Cleanup Script:	API_2_PRH_GCID_1.clr
Parameter File:	API_2_PRH_GCID_1.par

6.2.8.21.3 Test Procedure

Id	Description	API Expectation	APDU Expectation
0	`Applet1 is installed with maximum number of channel = 01.`		

1	**Channel status TLV is not present**
	
	`1- Build and send a DISPLAY TEXT command`
	
	`2- Call ProactiveResponseHandler.getChannelIdentifier() method.`
		2- UNAVAILABLE_ELEMENT ToolkitException is thrown	1- DISPLAY TEXT Proactive command is fetched.
			
			TERMINAL RESPONSE with no Channel status TLV available.

2	**Channel status TLV with a length equal to 0**
	`1- Build and send a OPEN CHANNEL proactive command`
	
	`2- Call ProactiveResponseHandler.getChannelIdentifier() method.`
		2- OUT_OF_TLV_BOUNDARIES ToolkitException is thrown	1- OPEN CHANNEL Proactive command is fetched.
			
			TERMINAL RESPONSE with Channel status TLV length equal to 0.

3	**Get channel identifier value**
	
	`1- Call ProactiveHandler.init() and ProactiveHandler.send() methods to open a channel.`
	
	`2- Call ProactiveResponseHandler.getChannelIdentifier() method.`
	
	`3- Call ProactiveHandler.initCloseChannel() and ProactiveHandler.send() methods.`
		2- Returns 0x01	1- OPEN CHANNEL Proactive Command is fetched.
			
			TERMINAL RESPONSE is issued with channel status value = 0x8100.

4	**Get channel identifier value with 2 TLV**
	
	`1- Call ProactiveHandler.init()and ProactiveHandler.send() methods to open a channel`
	
	`2- Call ProactiveResponseHandler.getChannelIdentifier()`
	
	`3- Call ProactiveHandler.initCloseChannel() and ProactiveHandler.send() methods.`
		2- Returns 0x01	1- OPEN CHANNEL Proactive Command is fetched.
			
			TERMINAL RESPONSE is issued with channel status value = 0x8100 and 0x8200.

5	**Channel status TLV is currently selected TLV**
	
	`1- Call ProactiveHandler.init() and ProactiveHandler.send() methods to open a channel.`
	`ViewHandler.FindTLV with Device Identity Tag.`
	
	`2- Call ProactiveResponseHandler.getChannelIdentifier() method.`
	
	`3- Compare ProactiveResponseHandler.getChannelIdentifier() and then ViewHandler.getValueByte(0) methods.`
		2- Returns 0x03
		
		3- Check getChannelIdentifier() =getValueByte(0)
			1- OPEN CHANNEL Proactive Command is fetched.
			
			TERMINAL RESPONSE is issued with channel status value = 0x0305~~.~~

6.2.8.21.4 Test Coverage

CRR number	Test case number
N1	3
N2	4
N3	5
C1	1

C2	2

6.2.8.22 Method copyChannelData

Test Area Reference: API_2_PRH_CCHD_BSS

6.2.8.22.1 Conformance Requirement:

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

```
public short copyChannelData(byte[] dstBuffer,
                             short dstOffset,
                             short dstLength)
                      throws java.lang.NullPointerException,
                             java.lang.ArrayIndexOutOfBoundsException,
                             ToolkitException
```

6.2.8.22.1 Normal execution

CRRN1: The method shall copy a part of the Channel data string field.

CRRN2: The Channel data string field value returned shall be the first Channel data TLV element of the current response data field.

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

CRRN4: Returns dstOffset + dstLength.

6.2.8.22.2 Parameters error

CRRP1: If dstBuffer is null, a NullPointerException is thrown.

CRRP2: If dstOffset or dstLength parameter is negative an ArrayIndexOutOfBoundsException exception is thrown and no copy is performed.

CRRP3: If dstOffset+dstLength is greater than dstBuffer.length, the length of the dstBuffer array an ArrayIndexOutOfBoundsException exception is thrown and no copy is performed.

CRRP4: If dstLength is greater than the value field of the available TLV, a OUT_OF_TLV_BOUNDARIES ToolkitException is thrown.

6.2.8.22.3 Context errors

CRRC1: The method shall throw a UNAVAILABLE_ELEMENT ToolkitException if the Result TLV is not present.

6.2.8.22.2 Test suite files

Test Script:	API_2_PRH_CCHD_BSS_1.scr
Test Applet:	API_2_PRH_CCHD_BSS_1.java
Load Script:	API_2_PRH_CCHD_BSS_1.ldr
Cleanup Script:	API_2_PRH_CCHD_BSS_1.clr
Parameter File:	API_2_PRH_CCHD_BSS_1.par

6.2.8.22.3 Test Procedure

Id	Description	API 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 one channel. ProactiveHandler.send() method is called.`
			2- OPEN CHANNEL proactive command is fetched
			
			TERMINAL RESPONSE is issued with Channel Id = 01

1	**CopyChannelData() with NULL dstBuffer**
	
	`Build and send a RECEIVE DATA command`
	
	`Call ProactiveResponseHandler.copyChannelData`
	`dstBuffer = NULL`
	`DstOffset = 0`
	`DstLength = 1`
		NullPointerException is thrown	RECEIVE DATA Proactive command is fetched.
			
			TERMINAL RESPONSE with not empty Channel Data TLV is issued.

2	**CopyChannelData() with negative dstOffset**
	
	`1- call init() method for the RECEIVE DATA proactive command.`
	
	`2- call ProactiveResponseHandler.copyChannelData()`
	`DstBuffer.length = 6`
	`DstOffset = -1`
	`DstLength = 1`
	
	`3- check dstBuffer is empty.`
		2- an ArrayIndexOutOfBoundsException exception is thrown.
		
		3- no copy is performed.
			1- RECEIVE DATA proactive command is fetched.
			
			TERMINAL RESPONSE with 6 bytes avalaible ('Hello1')

3	**CopyChannelData() with negative dstLength**
	
	`1- call ProactiveResponseHandler.copyChannelData()`
	`DstBuffer.length = 6`
	`DstOffset = 0`
	`DstLength = -1`
	
	`2- check dstBuffer is empty.`
		1- an ArrayIndexOutOfBoundsException exception is thrown.
		
		2- no copy is performed.	

4	**CopyChannelData() with dstOffset+dstLength greater than dstBuffer.length**
	
	`1- call ProactiveResponseHandler.copyChannelData() with dstOffset+dstLength greater than dstBuffer.length.`
	`DstBuffer.length = 6`
	`DstOffset = 5`
	`DstLength = 2`
	
	`2- check dstBuffer is empty.`
		1- an ArrayIndexOutOfBoundsException exception is thrown.
		
		2- no copy is performed.	

5	**CopyChannelData() with dstLength too large**
	
	`Call ProactiveResponseHandler.copyChannelData() with dstLength greater than the value field of the available TLV.`
	`DstBuffer.length = 6`
	`DstOffset = 0`
	`DstLength = 10`
		a OUT_OF_TLV_BOUNDARIES ToolkitException is thrown.	

6	**CopyChannelData() without Channel Data TLV element**
	
	`1- call init() method for the RECEIVE DATA proactive command.`
	`Call send() method.`
	
	`2- call ProactiveResponseHandler.copyChannelData()`
	`DstBuffer.length = 10`
	`DstOffset = 0`
		2- a UNAVAILABLE_ELEMENT ToolkitException is thrown.	1- RECEIVE DATA proactive command is fetched
			
			TERMINAL RESPONSE without ChannelData TLV element.

	DstLength = 10		

7	**Successful copyChannelData()**
	
	`Call init() method for the RECEIVE DATA proactive command.`
	`Call send() method.`
	
	`2- Call findTLV() with TAG of DEVICE IDENTITY.`
	
	`3- Call ProactiveResponseHandler.copyChannelData()`
	`DstBuffer.length = 6`
	`DstOffset = 0`
	`DstLength = 6`
	`DstBuffer is the whole Buffer.`
		3- the Channel Data TLV is copied into dstBuffer.
		
		The applet checks the returned value is dstOffset + dstLength = 6.
			1- RECEIVE DATA proactive command is fetched
			
			TERMINAL RESPONSE with one Channel data TLV element. (6 bytes available = 'Hello2')

8	**Compare copied Buffer** `Check dstBuffer.`	The applet checks that dstBuffer contains the channel data from the TERMINAL RESPONSE.	
9	**Check the Channel Data TLV is selected** `Call the ViewHandler.getValueByte(0) method`	The returned byte is the same than the first byte of the Channel data TLV (i.e. 'H')	

10	**Successful copyChannelData()**
	
	`Call ProactiveResponseHandler.copyChannelData()`
	`DstBuffer.length = 6`
	`DstOffset = 2`
	`DstLength = 3`
	
	`DstBuffer is a part of Buffer.`
		The Channel Data TLV is copied into dstBuffer.
		The applet checks the returned value is dstOffset + dstLength = 5.	

11	**Compare copied Buffer** `Check dstBuffer.`	The applet checks that bytes from 2 to 4 of dstBuffer contain the first 3 bytes of channel data TLV from the TERMINAL RESPONSE.	

12	**Successful copyChannelData()**
	
	`1- Initialise dstBuffer to [00, 01…]`
	
	`2- Call ProactiveResponseHandler.copyChannelData()`
	`DstBuffer.length = 6`
	`DstOffset = 2`
	`DstLength = 3`
	
	`DstBuffer is a part of buffer.`
		2- The Channel Data TLV is copied into dstBuffer.
		
		The returned value is dstOffset + dstLength = 5.	

13	**Compare copied Buffer** `Check dstBuffer.`	The applet checks that only bytes from 2 to 4 of dstBuffer have been updated with the first 3 bytes of channel data TLV from the TERMINAL RESPONSE.	

14	**Successful copyChannelData(), with 2 TLV**
	
	`1- call init() method for the RECEIVE DATA proactive command.`
	`Call send() method.`
	
	`2- call ProactiveResponseHandler.copyChannelData() with dstLength lower than the value field of the available TLV.`
	`DstBuffer.length = 6`
	`DstOffset = 0`
	`DstLength = 6`
		2- the first Channel Data TLV is copied into dstBuffer.
		The returned value is dstOffset+dstLength =0x06
			1- RECEIVE DATA proactive command is fetched
			
			TERMINAL RESPONSE with two Channel data TLV element
			1st TLV : 6 bytes available = 'Hello3'
			2nd TLV : 6 bytes available = 'Hello4'

15	**Compare copied Buffer**	Check that dstBuffer contains the	

	`Check dstBuffer.`	first Channel Data TLV from the TERMINAL RESPONSE.	

6.2.8.22.4 Test Coverage

CRR number	**Test case number**
N1	7, 10, 12, 14
N2	14
N3	9
N4	8, 11, 13, 15
P1	1
P2	2, 3
P3	4
P4	5
C1	6

6.2.9 Class ToolkitRegistry

6.2.9.3 Method clearEvent

Test Area Reference: API_2_TKR_CEVTB~~.~~

6.2.9.3.1 Conformance requirement:

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

```
public void clearEvent(byte event)
              throws ToolkitException,
                     javacard.framework.TransactionException
```

6.2.9.3.1.1 Normal execution

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

~~—~~CRRN2: The SIM Toolkit Framework shall not trigger the applet on the occurrence of the cleared event anymore.

CRRN3: if event was EVENT_CALL_CONTROL_BY_SIM and after the call, no applet is registered to it, The SIM Toolkit Framework shall allow an applet to register to this event.

CRRN4: if event was EVENT_CALL_CONTROL_BY_SIM and one applet is still registered to these event, The SIM Toolkit Framework shall not allow an applet to register to this event.

CRRN5: if event was EVENT_MO_SHORT_MESSAGE_CONTROL_BY_SIM and after the call, no applet is registered to it, The SIM Toolkit Framework shall allow an applet to register to this event.

~~—~~CRRN6: if event was EVENT_MO_SHORT_MESSAGE_CONTROL_BY_SIM and one applet is still registered to these event, The SIM Toolkit Framework shall not allow an applet to set this event.

6.2.9.3.1.2 Parameters error

CRRP1: Shall throw a Toolkit Exception with reason EVENT_NOT_ALLOWED if event was EVENT_MENU_SELECTION.

CRRP2: Shall throw a Toolkit Exception with reason EVENT_NOT_ALLOWED if event was EVENT_MENU_SELECTION_HELP_REQUEST.

CRRP3: Shall throw a Toolkit Exception with reason EVENT_NOT_ALLOWED if event was EVENT_TIMER_EXPIRATION.

CRRP4: Shall throw a Toolkit Exception with reason EVENT_NOT_ALLOWED if event was EVENT_STATUS_COMMAND.

6.2.9.3.1.3 ~~Context errors~~

~~No requirements.~~ Context errors

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

6.2.9.3.2 Test suite files

Test Script: API_2_TKR_CEVTB_1.scr

Test Applet: API_2_TKR_CEVTB_1.java

As default but applet registers to an event list which contains all defined events in ~~GSM 03.19~~ TS 43.019 [7] excepted those that aren't allowed or supported by setEvent().

Load ~~Script: API_2_TKR_CEVTB_1.ldr~~

~~Cleanup script: API_2_TKR_CEVTB_1.clr~~

~~Parameter File:~~Script: API_2_TKR_CEVTB_1.ldr

Cleanup script: API_2_TKR_CEVTB_1.clr

Parameter File: API_2_TKR_CEVTB_1.par

6.2.9.3.3 Test procedure

Id	Description	API Expectation	APDU Expectation
~~1~~	**~~Clear ALLOWED unregistered events~~** ~~For events ranging from 1 to 127 excepted those that aren't allowed (EVENT_MENU_SELECTION, EVENT_MENU_SELECTION_HELP_REQUEST, EVENT_TIMER_EXPIRATION, EVENT_STATUS_COMMAND), the applet calls:~~ ~~1- clearEvent() method~~ ~~2- isEventSet() method~~	~~1- No exception is thrown each time.~~ ~~2- Shall return false each time.~~	
1	**Clear ALLOWED unregistered events** For events ranging from -1, 1 to 24 and 127* excepted those that aren't allowed (7, 8, 11, 19), the applet calls: 1- clearEvent() method 2- isEventSet() method	1- No exception is thrown each time. 2- Shall return false each time.	
~~2~~	**~~Clear registered events~~** ~~1- For each ALLOWED and SUPPORTED events, the applet calls setEvent() method.~~ ~~2- For events ranging from 1 to 127 excepted those that aren't allowed, the applet calls:~~ ~~2.1- clearEvent() method~~ ~~2.2- isEventSet() method~~	~~1- No exception shall be thrown.~~ ~~2.1- No exception shall be thrown.~~ ~~2.2- Shall return false.~~	
2	**Clear registered events** 1- For each ALLOWED and SUPPORTED event (-1, 1 to 24 and 127)* excepted those that aren't allowed (7, 8, 11, 19), the applet calls setEvent() method. 2- For each ALLOWED and SUPPORTED event (-1, 1 to 24 and 127)* excepted those that aren't allowed (7, 8, 11, 19), the applet calls: 2.1- clearEvent() method 2.2- isEventSet() method	1- No exception shall be thrown. 2.1- No exception shall be thrown. 2.2- Shall return false.	
~~3~~	**~~Clearing NOT ALLOWED events~~** ~~For each event among: EVENT_MENU_SELECTION, EVENT_MENU_SELECTION_HELP_REQUEST, EVENT_TIMER_EXPIRATION, EVENT_STATUS_COMMAND~~ ~~1- The applet calls clearEvent(event) method.~~	~~1- Each time, clearEvent shall throw a Toolkit Exception with reason EVENT_NOT_ALLOWED.~~	
3	**Clearing NOT ALLOWED events** For each event among: EVENT_MENU_SELECTION, EVENT_MENU_SELECTION_HELP_REQUEST, EVENT_TIMER_EXPIRATION, EVENT_STATUS_COMMAND 1- The applet calls clearEvent(event) method.	1- Each time, clearEvent shall throw a ToolkitException with reason EVENT_NOT_ALLOWED.	

~~4~~	~~**Checking applet isn't triggered by an ENVELOPE(SMS-PP DOWNLOAD) command**~~ ~~`1 - reset and initialize the card`~~ ~~`2 - An ENVELOPE(SMS-PP DOWNLOAD) is sent with a TAR referencing applet.`~~	~~Applet is not trigged by an ENVELOPE(SMS-PP DOWNLOAD) command~~	

4	**Checking applet isn't triggered by an ENVELOPE(SMS-PP DOWNLOAD) command**
	`1 - reset and initialise the card`
	`2 - An ENVELOPE(SMS-PP DOWNLOAD) is sent with a TAR referencing applet.`
		Applet is not trigged by an ENVELOPE(SMS-PP DOWNLOAD) command	

*Note: Although the method clearEvent is defined for a range from –128 to 127 only the allowed events are tested here, because the range from -128 to –2 is reserved for propriatary use in TS 43.019 [7] chapter 6.2 and the range from 25 to 126 is omitted for compatibility with future releases of TS 43.019 [7]

6.2.9.3.4 Test Coverage

CRR number	Test case number
~~N1~~	~~1, 2~~
N1	1,2
N2	4
N3	Framework
N4	Framework
N5	Framework
N6	Framework
P1	3
P2	3
P3	3
P4	3
C1	not testable

6.2.9.9 Method isEventSet

Test Area Reference: API_2_TKR_IEVSB~~.~~

6.2.9.9.1 Conformance requirement:

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

```
public boolean isEventSet(byte event)
```

6.2.9.9.1.1 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 isn't set in the Toolkit Registry for the applet~~.~~

~~6.2.9.9.1.2 Parameters error~~

~~No requirements.~~

~~6.2.9.9.1.3 Context errors~~

~~No requirements.~~

6.2.9.9.2 Test suite files

Test Script: API_2_TKR_IEVSB_1.scr

Test Applet: ~~API_2_TKR_IEVSB_1.java~~

~~Installation parameter:~~ API_2_TKR_IEVSB_1.java

API_2_TKR_IEVSB_2.java

Installation parameter:

Same as default applet but with:

- Maximum text length for a menu entry: 15
- Maximum number of menu entries: 1
- Position / Identifier for each menu entry: '01'/'01'
- Maximum number of timers: 1

Load Script:	API_2_TKR_IEVSB_1.ldr
Cleanup script:	API_2_TKR_IEVSB_1.clr
Parameter File:	API_2_TKR_IEVSB_1.par

6.2.9.9.3 Test procedure

Id	Description	API Expectation	APDU Expectation
~~1~~	~~**Events aren't set**~~ ~~`Applet calls isEventSet() for each events ranging from 1 to 127 excepted EVENT_FORMATTED_SMS_PP_ENV and EVENT_MENU_SELECTION.`~~	~~Shall return false each time.~~	
1	**Install Applet1 only registered to EVENT FORMATTED_SMS_PP_ENV and EVENT_MENU_SELECTION** **Test that events aren't set** `Applet calls isEventSet() for each event ranging from –1, 1 to 24 and 127* excepted EVENT_FORMATTED_SMS_PP_ENV (2) and EVENT_MENU_SELECTION (7).`	Shall return false each time.	
~~2~~	~~**For EVENT_FORMATTED_SMS_PP_ENV**~~ ~~`isEventSet (EVENT_FORMATTED_SMS_PP_ENV)`~~	~~Shall return true.~~	
2	**For EVENT_FORMATTED_SMS_PP_ENV** `isEventSet(EVENT_FORMATTED_SMS_PP_ENV)`	Shall return true.	
~~3~~	~~**For EVENT_MENU_SELECTION**~~ ~~`isEventSet (EVENT_MENU_SELECTION)`~~	~~Shall return true~~	
3	**For EVENT_MENU_SELECTION** `isEventSet(EVENT_MENU_SELECTION)`	Shall return true	
~~4~~	~~**After clearing EVENT_FORMATTED_SMS_PP_ENV**~~ ~~`1- clearEvent(EVENT_FORMATTED_SMS_PP_ENV)`~~ ~~`2- isEventSet(EVENT_FORMATTED_SMS_PP_ENV).`~~	~~1- No exception shall be thrown.~~ ~~2- Shall return false.~~	

4	**After clearing EVENT_FORMATTED_SMS_PP_ENV**
	
	`1- clearEvent(EVENT_FORMATTED_SMS_PP_ENV)`
	
	`2- isEventSet(EVENT_FORMATTED_SMS_PP_ENV)`
		1- No exception shall be thrown.
		
		2- Shall return false.	

~~5~~	~~**Setting events**~~ ~~`1- For each SUPPORTED and ALLOWED events for setEvent(), applet calls:`~~ ~~`1.1- setEvent() method`~~ ~~`1.2- isEventSet() method.`~~	~~1.1- No exception shall be thrown.~~ ~~1.2- Shall return true each time.~~	
5	**Setting events** `For all allowed events defined in TS 43.019–[7] for method setEvent(): EVENT_PROFILE_DOWNLOAD, EVENT_FORMATTED_SMS_PP_ENV, EVENT_FORMATTED_SMS_PP_UPD, EVENT_FORMATTED_SMS_CB, EVENT_UNFORMATTED_SMS_PP_ENV, EVENT_UNFORMATTED_SMS_PP_UPD, EVENT_UNFORMATTED_SMS_CB,`	1- No exception shall be thrown. 2- Shall return true each time.	

	EVENT_CALL_CONTROL_BY_SIM, EVENT_MO_SHORT_MESSAGE_CONTROL_BY_SIM, EVENT_EVENT_DOWNLOAD_MT_CALL, EVENT_EVENT_DOWNLOAD_CALL_CONNECTED, EVENT_EVENT_DOWNLOAD_CALL_DISCONNECTED, EVENT_EVENT_DOWNLOAD_LOCATION_STATUS, EVENT_EVENT_DOWNLOAD_USER_ACTIVITY, EVENT_EVENT_DOWNLOAD_IDLE_SCREEN_AVAILABLE , EVENT_EVENT_DOWNLOAD_CARD_READER_STATUS, EVENT_EVENT_DOWNLOAD_LANGUAGE_SELECTION, EVENT_EVENT_DOWNLOAD_BROWSER_TERMINATION, EVENT_EVENT_DOWNLOAD_DATA_AVAILABLE, EVENT_EVENT_DOWNLOAD_CHANNEL_STATUS, EVENT_FIRST_COMMAND_AFTER_SELECT, EVENT_UNRECOGNIZED_ENVELOPE applet calls: 1- setEvent() method 2- isEventSet() method		
~~6~~	**~~For EVENT_MENU_SELECTION_HELP_REQUEST~~** ~~1- isEventSet (EVENT_MENU_SELECTION_HELP_REQUEST)~~ ~~2- call changeMenuEntry with help supported~~ ~~3- isEventSet (EVENT_MENU_SELECTION_HELP_REQUEST)~~	~~1- Shall return false.~~ ~~3- Shall return true~~	
6	**For EVENT_MENU_SELECTION_HELP_REQUEST** 1- isEventSet(EVENT_MENU_SELECTION_HELP_REQUEST) 2- call changeMenuEntry() with help supported 3- isEventSet(EVENT_MENU_SELECTION_HELP_REQUEST)	1- Shall return false. 3- Shall return true.	
~~7~~	**~~For EVENT_TIMER_EXPIRATION~~** ~~1- isEventSet(EVENT_TIMER_EXPIRATION)~~ ~~2- call allocateTimer()~~ ~~3- isEventSet(EVENT_TIMER_EXPIRATION)~~	~~1- Shall return false.~~ ~~3- Shall return true~~	
7	**For EVENT_TIMER_EXPIRATION** 1- isEventSet(EVENT_TIMER_EXPIRATION) 2- call allocateTimer() 3- isEventSet(EVENT_TIMER_EXPIRATION)	1- Shall return false. 3- Shall return true.	
~~8~~	**~~For EVENT_STATUS_COMMAND~~** ~~1- isEventSet(EVENT_STATUS_COMMAND)~~ ~~2- call requestPollInterval(POLL_SYSTEM_DURATION)~~ ~~3- isEventSet(EVENT_STATUS_COMMAND)~~	~~1- Shall return false.~~ ~~3- Shall return true~~	
8	**For EVENT_STATUS_COMMAND** 1- isEventSet(EVENT_STATUS_COMMAND) 2- call requestPollInterval(POLL_SYSTEM_DURATION) 3- isEventSet(EVENT_STATUS_COMMAND)	1- Shall return false. 3- Shall return true.	
9	**Install Applet2 only registered to EVENT_FORMATTED_SMS_PP_ENV** isEventSet(EVENT_MENU_SELECTION)	Shall return false.	

*Note: Although the method isEventSet() is defined for a range from –128 to 127 only the allowed events are tested, because the range from -128 to –2 is reserved for propriatary use in TS 43.019 [7] chapter 6.2 and the range from 25 to 126 is omitted for compatibility with future releases of TS 43.019 [7]

6.2.9.9.4 Test Coverage

CRR number	Test case number
~~N1~~	~~2, 3, 4, 5, 6, 7, 8~~
~~N2~~	~~1, 5, 6, 7, 8~~
N1	2,3,4,5,6,7,8
N2	1,5,6,7,8,9

6.2.9.12 Method setEvent

Test Area Reference: API_2_TKR_SEVTB~~.~~

6.2.9.12.1 Conformance Requirement:

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

```
public void setEvent(byte id)
        throws ToolkitException,
               javacard.framework.TransactionException
```

6.2.9.12.1.1 Normal execution

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

CRRN2: the SIM Toolkit Framework shall trigger the applet if an occurrence of the set event happens.

CRRN3: ~~this~~the method shall accept all the events defined in ~~GSM 0319 excepted:~~TS 43.019 [7] except: EVENT_MENU_SELECTION, EVENT_MENU_SELECTION_HELP_REQUEST, EVENT_TIMER_EXPIRATION, EVENT_STATUS_COMMAND.

CRRN4: no exception shall be thrown if the applet registers more than once to the same event.

CRRN5: all updates in the ToolkitRegistry are atomic.

6.2.9.12.1.2 Parameters error

CRRP1: shall throw a ToolkitException with EVENT_NOT_SUPPORTED reason if event is 0.

CRRP2: shall throw a ToolkitException with EVENT_NOT_ALLOWED reason if event is EVENT_MENU_SELECTION.

CRRP3: shall throw a ToolkitException with EVENT_NOT_ALLOWED reason if event is EVENT_MENU_SELECTION_HELP_REQUEST.

CRRP4: shall throw a ToolkitException with EVENT_NOT_ALLOWED reason if event is EVENT_TIMER_EXPIRATION.

CRRP5: shall throw a ToolkitException with EVENT_NOT_ALLOWED reason if event is EVENT_STATUS_COMMAND.

6.2.9.12.1.3 Context errors

CRRC1: shall throw a ToolkitException with EVENT_ALREADY_REGISTERED if event is EVENT_CALL_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 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_ENV and the applet has no TAR defined.

CRRC6: shall throw javacard.framework.TransactionException - if the operation would cause the commit capacity to be exceeded.

6.2.9.12.2 Test suite files

Test Script:	API_2_TKR_SEVTB_1.scr
Test Applet:	~~API_2_TKR_SEVTB_1.java~~
	~~API_2_TKR_SEVTB_2.java~~
~~Load Script:~~	API_2_TKR_SEVTB_1.java
	API_2_TKR_SEVTB_2.java
	API_2_TKR_SEVTB_3.java
	API_2_TKR_SEVTB_4.java
Load Script:	API_2_TKR_SEVTB_1.ldr
	The load script installs the ~~2~~4 instances.
Cleanup script:	API_2_TKR_SEVTB_1.clr
Parameter File:	API_2_TKR_SEVTB_1.par

6.2.9.12.3 Test Procedure

Id	Description	API Expectation	APDU Expectation
~~1~~	~~**Applet 1 is triggered by ENVELOPE(SMS_PP_FORMATTED) command.**~~ ~~`Send ENVELOPE(SMS_PP_FORMATTED)`~~	~~Applet 1 shall be triggered~~	
1	**Applet 1 is triggered by ENVELOPE(SMS_PP_FORMATTED) command.** `Send ENVELOPE(SMS_PP_FORMATTED)`	Applet 1 shall be triggered	
~~2~~	~~**Setting ALLOWED and SUPPORTED events**~~ ~~`1- For all events defined in GSM 0319 (from 1 to 19) and allowed: EVENT_PROFILE_DOWNLOAD, EVENT_FORMATTED_SMS_PP_ENV, EVENT_FORMATTED_SMS_PP_UPD, EVENT_UNFORMATTED_SMS_PP_ENV, EVENT_UNFORMATTED_SMS_PP_UPD, EVENT_FORMATTED_SMS_CB, EVENT_UNFORMATTED_SMS_CB, EVENT_CALL_CONTROL_BY_SIM, EVENT_MO_SHORT_MESSAGE_CONTROL_BY_SIM, EVENT_EVENT_DOWNLOAD_MT_CALL, EVENT_EVENT_DOWNLOAD_CALL_CONNECTED, EVENT_EVENT_DOWNLOAD_CALL_DISCONNECTED, EVENT_EVENT_DOWNLOAD_LOCATION_STATUS, EVENT_EVENT_DOWNLOAD_USER_ACTIVITY, EVENT_EVENT_DOWNLOAD_IDLE_SCREEN_AVAILABLE, EVENT_EVENT_DOWNLOAD_CARD_READER_STATUS`~~	~~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.~~	

	~~EVENT_EVENT_DOWNLOAD_LANGUAGE_SELECTION, EVENT_EVENT_DOWNLOAD_BROWSER_TERMINATION~~ ~~.~~ ~~1.1- clearEvent(event)~~ ~~1.2- isEventSet(event)~~ ~~1.3- setEvent(event)~~ ~~1.4- isEventSet(event)~~ ~~1.5- clearEvent(event)~~		

2	**Setting ALLOWED and SUPPORTED events**
	`1- For all allowed events (-1, 1 to 24 and 127 excepted 7, 8, 11, 19) defined in TS 43.019 [7]*: EVENT_PROFILE_DOWNLOAD, EVENT_FORMATTED_SMS_PP_ENV, EVENT_FORMATTED_SMS_PP_UPD, EVENT_FORMATTED_SMS_CB, EVENT_UNFORMATTED_SMS_PP_ENV, EVENT_UNFORMATTED_SMS_PP_UPD, EVENT_UNFORMATTED_SMS_CB, EVENT_CALL_CONTROL_BY_SIM, EVENT_MO_SHORT_MESSAGE_CONTROL_BY_SIM, EVENT_EVENT_DOWNLOAD_MT_CALL, EVENT_EVENT_DOWNLOAD_CALL_CONNECTED, EVENT_EVENT_DOWNLOAD_CALL_DISCONNECTED, EVENT_EVENT_DOWNLOAD_LOCATION_STATUS, EVENT_EVENT_DOWNLOAD_USER_ACTIVITY, EVENT_EVENT_DOWNLOAD_IDLE_SCREEN_AVAILABLE, EVENT_EVENT_DOWNLOAD_CARD_READER_STATUS, EVENT_EVENT_DOWNLOAD_LANGUAGE_SELECTION, EVENT_EVENT_DOWNLOAD_BROWSER_TERMINATION , EVENT_EVENT_DOWNLOAD_DATA_AVAILABLE, EVENT_EVENT_DOWNLOAD_CHANNEL_STATUS, EVENT_FIRST_COMMAND_AFTER_SELECT, EVENT_UNRECOGNIZED_ENVELOPE`
	`1.1- clearEvent(event)`
	`1.2- isEventSet(event)`
	`1.3- setEvent(event)`
	`1.4- isEventSet(event)`
	`1.5- 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.	

3	**Event 0** `Call setEvent(0)`	Shall throw a ToolkitException with EVENT_NOT_SUPPORTED reason code.	
4	**Setting EVENT_MENU_SELECTION** `Call setEvent(EVENT_MENU_SELECTION)`	Shall throw a ToolkitException with EVENT_NOT_ALLOWED reason code.	
5	**Setting EVENT_MENU_SELECTION_HELP_REQUEST** `Call setEvent(EVENT_MENU_SELECTION_HELP_REQUEST)`	Shall throw a ToolkitException with EVENT_NOT_ALLOWED reason code.	
6	**Setting EVENT_TIMER_EXPIRATION** `Call setEvent(EVENT_TIMER_EXPIRATION)`	Shall throw a ToolkitException with EVENT_NOT_ALLOWED reason	

		code.	
7	**Setting EVENT_STATUS_COMMAND** `Call setEvent(EVENT_STATUS_COMMAND)`	Shall throw a ToolkitException with EVENT_NOT_ALLOWED reason code.	
8	**Setting EVENT_CALL_CONTROL_BY_SIM** `Call setEvent(EVENT_CALL_CONTROL_BY_SIM)`	No Exception shall be thrown	
9	**Setting EVENT_MO_SHORT_MESSAGE_CONTROL_BY_SIM** `Call setEvent(EVENT_MO_SHORT_MESSAGE_ CONTROL_BY_SIM)`	No Exception shall be thrown	
10	**Check applet is triggered by an ENVELOPE(CALL_CONTROL_BY_SIM)** Trigger the applet	Applet is triggered by an ENVELOPE(CALL_CONTROL_BY_SIM)	
11	**Check applet is triggered by an ENVELOPE(MO_SHORT_MESSAGE_CONTROL_BY_SIM)** Trigger the Applet	Applet is triggered by an ENVELOPE(MO_SHORT_MESSAGE_CONTROL_BY_SIM)	
12	**Applet 2 is triggered by ENVELOPE(SMS_ PP_DOWNLOAD) command.** `Trigger the applet 2`	Applet 2 is triggered by an ENVELOPE(SMS_ PP_DOWNLOAD) command	
13	**Applet 2 registers to CALL_CONTROL_BY_SIM but it is already assigned** `setEvent(EVENT_CALL_CONTROL_BY_SIM)`	Shall throw a ToolkitException with EVENT_ALREADY_REGISTERED reason code.	
14	**Applet 2 registers to MO_MESSAGE_CONTROL_BY SIM but it is already assigned** `setEvent(EVENT_MO_SHORT_MESSAGE_CONTROL_ BY_SIM)`	Shall throw a ToolkitException with EVENT_ALREADY_REGISTERED reason code.	

15	**Applet 3 with no TAR defined registers to EVENT_UNFORMATTED_SMS_CB**
	`1- send ENVELOPE(CELL_BROADCAST_DATA_ DOWNLOAD)`
	`2- setEvent(FORMATTED_SMS_PP_ENV)`
	`3- setEvent(FORMATTED_SMS_PP_UPD)`
	`4- setEvent(FORMATTED_SMS_CB_ENV)`
		1- Applet 3 shall be triggered
		2- ToolkitException with reason code TAR_NOT_DEFINED should be thrown
		3- ToolkitException with reason code TAR_NOT_DEFINED should be thrown
		4- ToolkitException with reason code TAR_NOT_DEFINED should be thrown	

16	**Applet 4 registers multiple to EVENT_FORMATTED_SMS_PP_ENV**
	`1- send ENVELOPE(EVENT_FORMATTED_ SMS_PP_ENV)`
	`2- setEvent(EVENT_FORMATTED_SMS_PP_ UPD)`
	`3- setEvent(EVENT_FORMATTED_SMS_PP_ UPD)`
	`4- send ENVELOPE(EVENT_FORMATTED_ SMS_PP_UPD)`
		1- Applet 4 shall be triggered
		2- no Exception shall be thrown
		3- no Exception shall be thrown
		4- Applet 4 shall be triggered	

*Note: Although the method setEvent is defined for a range from –128 to 127 only the allowed events are tested, because the range from -128 to –2 is reserved for propriatary use in TS TS 43.019 [7] chapter 6.2 and the range from 25 to 126 is omitted for compatibility with future releases of TS 43.019 [7]

6.2.9.12.4 Test Coverage

CRR number	Test case number
N1	2
~~N2~~	~~1, 8, 9, 10, 11, 12~~
N2	1, 8,9,10, 11, 12
~~N3~~	~~2, 4, 5, 6, 7~~
N3	2,4,5,6,7
N4	16
N5	not testable
P1	3
P2	4
P3	5
P4	6
P5	7
C1	13
C2	14
C3	15
C4	15
C5	15
C6	not testable

6.2.9.13 Method setEventList

Test Area Reference: API_2_TKR_SEVL_BSS~~.~~

6.2.9.13.1 Conformance Requirement:

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

```
public void setEventList(byte[] eventList,
                         short offset,
                         short length)
                throws   java.lang.NullPointerException,
                         java.lang.ArrayIndexOutOfBoundsException,
                         ToolkitException,
                         javacard.framework.TransactionException
```

6.2.9.13.1.1 Normal execution

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

CRRN2: the SIM Toolkit Framework shall trigger the applet if an occurrence of one of the successfully registered events happens.

~~–~~CRRN3: this method shall accept all the events defined in ~~GSM 0319 excepted:~~TS 43.019 [7] except: EVENT_MENU_SELECTION, EVENT_MENU_SELECTION_HELP_REQUEST, EVENT_TIMER_EXPIRATION , EVENT_STATUS_COMMAND.

CRRN4: all updates on the ToolkitRegistry are atomic

CRRN5: No exception shall be thrown if the applet registers more than once to the same event.

6.2.9.13.1.2 Parameters error

CRRP1: shall throw a java.lang.NullPointerException if eventList is null.

CRRP2: shall throw a java.lang.ArrayIndexOutOfBoundsException if offset would cause access outside array bounds.

CRRP3: shall throw a java.lang.ArrayIndexOutOfBoundsException if length would cause access outside array bounds.

CRRP4: shall throw a java.lang.ArrayIndexOutOfBoundsException if both offset and length would cause access outside array bounds.

CRRP5: shall throw a ToolkitException with EVENT_NOT_SUPPORTED reason if event is 0.

CRRP6: shall throw a ToolkitException with EVENT_NOT_ALLOWED reason if eventList contains EVENT_MENU_SELECTION.

CRRP7: shall throw a ToolkitException with EVENT_NOT_ALLOWED reason if eventList contains EVENT_MENU_SELECTION_HELP_REQUEST.

CRRP8: shall throw a ToolkitException with EVENT_NOT_ALLOWED reason if eventList contains EVENT_TIMER_EXPIRATION.

CRRP9: shall throw a ToolkitException with EVENT_NOT_ALLOWED reason if eventList contains EVENT_STATUS_COMMAND.

6.2.9.13.1.3 Context errors

CRRC1: shall throw a ToolkitException with EVENT_ALREADY_REGISTERED if eventList contains EVENT_CALL_CONTROL_BY_SIM but another applet is already registered to it.

CRRC2: 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.

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_ENV and the applet has no TAR defined.

CRRC6: shall throw javacard.framework.TransactionException - if the operation would cause the commit capacity to be exceeded.

6.2.9.13.2 Test suite files

Test Script:	API_2_TKR_SEVL_BSS_1.scr
Test Applet:	~~API_2_TKR_SEVL_BSS_1.java~~
	~~API_2_TKR_SEVL_BSS_2.java~~
~~Load Script:~~	API_2_TKR_SEVL_BSS_1.java
	API_2_TKR_SEVL_BSS_2.java
	API_2_TKR_SEVL_BSS _3.java
Load Script:	API_2_TKR_SEVL_BSS_1.ldr
	The load script installs the ~~2~~4 instances.
Cleanup script:	API_2_TKR_SEVL_BSS_1.clr
Parameter File:	API_2_TKR_SEVL_BSS_1.par

6.2.9.13.3 Test Procedure

Id	Description	API Expectation	APDU Expectation
~~1~~	**~~Applet 1 Registering all eventList buffer~~** ~~EventList = all allowed events defined in GSM 0319:~~ ~~EVENT_PROFILE_DOWNLOAD,~~ ~~EVENT_FORMATTED_SMS_PP_ENV,~~ ~~EVENT_FORMATTED_SMS_PP_UPD,~~ ~~EVENT_UNFORMATTED_SMS_PP_ENV,~~ ~~EVENT_UNFORMATTED_SMS_PP_UPD,~~ ~~EVENT_FORMATTED_SMS_CB,~~ ~~EVENT_UNFORMATTED_SMS_CB,~~ ~~EVENT_CALL_CONTROL_BY_SIM,~~ ~~EVENT_MO_SHORT_MESSAGE_CONTROL_BY_SIM,~~ ~~EVENT_EVENT_DOWNLOAD_MT_CALL,~~ ~~EVENT_EVENT_DOWNLOAD_CALL_CONNECTED,~~ ~~EVENT_EVENT_DOWNLOAD_CALL_DISCONNECTED,~~ ~~EVENT_EVENT_DOWNLOAD_LOCATION_STATUS,~~ ~~EVENT_EVENT_DOWNLOAD_USER_ACTIVITY,~~ ~~EVENT_EVENT_DOWNLOAD_IDLE_SCREEN_AVAILABLE, EVENT_EVENT_DOWNLOAD_CARD_READER_STATUS~~ ~~EVENT_EVENT_DOWNLOAD_LANGUAGE_SELECTION,~~ ~~EVENT_EVENT_DOWNLOAD_BROWSER_TERMINATION.~~ ~~1- For each event in EventList clearEvent(event)~~ ~~2- setEventList(eventList)~~ ~~Offset = 0~~ ~~Length = eventList.lentgh~~ ~~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.~~	
1	**Applet 1 Registering all eventList buffer** EventList = all allowed events (-1, 1 to 24 and 127 excepted 7, 8, 11, 19) defined in TS 43.019~~-~~[7]: EVENT_PROFILE_DOWNLOAD, EVENT_FORMATTED_SMS_PP_ENV, EVENT_FORMATTED_SMS_PP_UPD, EVENT_FORMATTED_SMS_CB, EVENT_UNFORMATTED_SMS_PP_ENV, EVENT_UNFORMATTED_SMS_PP_UPD, EVENT_UNFORMATTED_SMS_CB, EVENT_CALL_CONTROL_BY_SIM, EVENT_MO_SHORT_MESSAGE_CONTROL_BY_SIM, EVENT_EVENT_DOWNLOAD_MT_CALL, EVENT_EVENT_DOWNLOAD_CALL_CONNECTED, EVENT_EVENT_DOWNLOAD_CALL_DISCONNECTED, EVENT_EVENT_DOWNLOAD_LOCATION_STATUS, EVENT_EVENT_DOWNLOAD_USER_ACTIVITY, EVENT_EVENT_DOWNLOAD_IDLE_SCREEN_AVAILABLE, EVENT_EVENT_DOWNLOAD_CARD_READER_STATUS, EVENT_EVENT_DOWNLOAD_LANGUAGE_SELECTION, EVENT_EVENT_DOWNLOAD_BROWSER_TERMINATION, EVENT_EVENT_DOWNLOAD_DATA_AVAILABLE, EVENT_EVENT_DOWNLOAD_CHANNEL_STATUS, EVENT_FIRST_COMMAND_AFTER_SELECT, EVENT_UNRECOGNIZED_ENVELOPE 1- For each event in EventList clearEvent(event) 2- setEventList(eventList) Offset = 0 Length = eventList.lentgh	1- No exception shall be thrown. 2- No exception shall be thrown. 3- Each time shall return true.	

	3- For all events in eventList isEventSet(event) 4- For each event in EventList clearEvent(event)	4- No exception shall be thrown.	
~~2~~	~~**Registering part of eventList buffer**~~ ~~EventList = all allowed events defined in GSM 0319 (see test case 1).~~ ~~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:~~ ~~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.~~	
2	**Registering part of eventList buffer** EventList = all allowed events defined in TS 43.019~~-~~[7] (see test case 1). 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: 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	**Null buffer** `EventList = null`	Shall throw a java.lang.NullPointerException Exception	

4	**Out of bounds offset**
	
	`Offset = eventList.length`
	`Length = 1`
		Shall throw a java.lang.ArrayIndexOutOfBounds Exception	

5	**Out of bounds and big offset**
	
	`Offset = 255`
	`Length = 1`
		Shall throw a java.lang.ArrayIndexOutOfBounds Exception	

6	**Offset < 0**
	
	`Offset = -1`
	`Length = 1`
		Shall throw a java.lang.ArrayIndexOutOfBounds Exception	

7	**Out of bounds length**
	
	`Offset = 0`
	`Length = eventList.length + 1`
		Shall throw a java.lang.ArrayIndexOutOfBounds Exception	

8	**Out of bounds and big length**	Shall throw a	

	`Offset = 0`
	`Length = 255`
		java.lang.ArrayIndexOutOfBounds Exception	

9	**Length < 0**
	`Offset = 0`
	`Length = -1`
		Shall throw a java.lang.ArrayIndexOutOfBounds Exception	

10	**Out of bounds offset + Length** `Offset + length > eventList.length + 1`	Shall throw a java.lang.ArrayIndexOutOfBounds Exception	
11	**Event 0** `Call setEventList(eventList) with eventList indicating event 0`	Shall throw a ToolkitException with EVENT_NOT_SUPPORTED reason code.	
12	**EVENT_MENU_SELECTION** `Call setEventList(eventList) with eventList indicating EVENT_MENU_SELECTION`	Shall throw a ToolkitException with reason code EVENT_NOT_ALLOWED.	
13	**EVENT_MENU_SELECTION_HELP_REQUEST** `Call setEventList(eventList) with eventList indicating EVENT_MENU_SELECTION_HELP_REQUEST`	Shall throw a ToolkitException with reason code EVENT_NOT_ALLOWED.	
14	**EVENT_TIMER_EXPIRATION** `Call setEventList(eventList) with eventList indicating EVENT_TIMER_EXPIRATION`	Shall throw a ToolkitException with reason code EVENT_NOT_ALLOWED.	
15	**EVENT_STATUS_COMMAND** `Call setEventList(eventList) with eventList indicating EVENT_STATUS_COMMAND`	Shall throw a ToolkitException with reason code EVENT_NOT_ALLOWED.	
16	**Setting EVENT_CALL_CONTROL_BY_SIM** `setEventList(List, 0, 2) with List containing EVENT_CALL_CONTROL_BY_SIM & EVENT_MO_SHORT_MESSAGE_CONTROL_BY_SIM`	Shall not throw an exception	
~~17~~	~~**Check applet is triggered by an ENVELOPE(CALL_CONTROL_BY_SIM)**~~ ~~`Reset and initialize the card`~~ ~~`Trigger the applet`~~	~~Applet is trigged by an ENVELOPE(CALL_CONTROL_BY_SIM)~~	

17	**Check applet is triggered by an ENVELOPE(CALL_CONTROL_BY_SIM)**
	`Reset and initialise the card`
	`Trigger the applet`
		Applet is trigged by an ENVELOPE(CALL_CONTROL_BY_SIM)	

18	**Check applet is triggered by an ENVELOPE(MO_SHORT_MESSAGE_CONTROL_BY_SIM)** `Trigger the applet`	Applet is trigged by an ENVELOPE(MO_SHORT_MESSAGE_CONTROL_BY_SIM)	
19	**Applet 2 registers to CALL_CONTROL_BY_SIM but it is already assigned** `setEventList(MonoEventList,0,1) with MonoEventList containing EVENT_CALL_CONTROL_BY_SIM`	Shall throw a ToolkitException with EVENT_ALREADY_REGISTERED reason code.	
20	**Applet 2 registers to MO_SHORT_MESSAGE_CONTROL_BY_SIM but it is already assigned setEventList(MonoEventList,0,1) with MonoEventList containing EVENT_MO_SHORT_MESSAGE_CONTROL_BY_SIM**	Shall throw a ToolkitException with EVENT_ALREADY_REGISTERED reason code.	

21	**Applet 3 with no TAR defined registers to EVENT_UNFORMATTED_SMS_CB** 1- send ENVELOPE(EVENT_UNFORMATTED_SMS_CB) 2- setEventList(EVENT_FORMATTED_SMS_PP_ENV, EVENT_UNFORMATTED_SMS_PP_ENV, EVENT_UNFORMATTED_SMS_PP_ENV) 3- setEventList(EVENT_UNFORMATTED_SMS_PP_ENV, EVENT_FORMATTED_SMS_PP_UPD, EVENT_UNFORMATTED_SMS_PP_ENV) 4- setEventList(EVENT_UNFORMATTED_SMS_PP_ENV, EVENT_UNFORMATTED_SMS_PP_ENV, EVENT_FORMATTED_SMS_CB_ENV) 5- isEventSet(EVENT_UNFORMATTED_SMS_PP_ENV) 6- isEventSet(EVENT_UNFORMATTED_SMS_PP_UPD) 7- isEventSet(EVENT_FORMATTED_SMS_PP_ENV) 8- isEventSet(EVENT_FORMATTED_SMS_PP_UPD) 9- isEventSet(EVENT_FORMATTED_SMS_CB_ENV)	1- Applet3 shall be triggered 2- ToolkitException with reason code TAR_NOT_DEFINED should be thrown 3- ToolkitException with reason code TAR_NOT_DEFINED should be thrown 4- ToolkitException with reason code TAR_NOT_DEFINED should be thrown 5- method should return FALSE 6- method should return FALSE 7- method should return FALSE 8- method should return FALSE 9- method should return FALSE	
22	1- setEventList(EVENT_UNFORMATTED_SMS_PP_ENV, EVENT_UNFORMATTED_SMS_PP_ENV) 2- isEventSet(EVENT_UNFORMATTED_SMS_PP_ENV)	1- no exception should be thrown 2- method should return true	

6.2.9.13.4 Test Coverage

CRR number	Test case number
N1	1,2
N2	16,17,18
N3	1,2,11,12,13,14,15
N4	21
N5	22
P1	3
P2	4,5,6
P3	7,8,9
P4	10
P5	11
P6	12
P7	13
P8	14
P9	15
C1	19
C2	20
C3	21
C4	21
C5	21
C6	not testable

6.3 SIM Toolkit Framework

6.3.1 Minimum Handler Availability

This test area tests the rules that define the minimum requirements for the availability of the system handlers.

6.3.1.1 ProactiveHandler

Test Area Reference: FWK_MHA_PAHD~~.~~

6.3.1.1.1 Conformance Requirement

6.3.1.1.1.1 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_UNFORMATTED_SMS_PP_ENV

EVENT_FORMATTED_SMS_PP_UPD

EVENT_UNFORMATTED_SMS_PP_UPD

EVENT_FORMATTED_SMS_CB

EVENT_UNFORMATTED_SMS_CB

EVENT_MENU_SELECTION

EVENT_MENU_SELECTION_HELP_REQUEST

EVENT_TIMER_EXPIRATION

EVENT_EVENT_DOWNLOAD_MT_CALL

EVENT_EVENT_DOWNLOAD_CALL_CONNECTED

EVENT_EVENT_DOWNLOAD_CALL_DISCONNECTED

EVENT_EVENT_DOWNLOAD_LOCATION_STATUS

EVENT_EVENT_DOWNLOAD_USER_ACTIVITY

EVENT_EVENT_DOWNLOAD_IDLE_SCREEN_AVAILABLE

EVENT_EVENT_DOWNLOAD_CARD_READER_STATUS

EVENT_EVENT_DOWNLOAD_LANGUAGE_SELECTION

EVENT_EVENT_DOWNLOAD_BROWSER_TERMINATION

EVENT_UNRECOGNIZED_ENVELOPE

EVENT_STATUS_COMMAND

EVENT_CALL_CONTROL

EVENT_SMS_MO_CONTROL

EVENT_PROFILE_DOWNLOAD

~~6.3.1.1.1.2 Parameters error~~

~~No requirements.~~

~~6.3.1.1.1.3 Context errors~~

~~No requirements.~~EVENT_EVENT_DOWNLOAD_DATA_AVAILABLE

EVENT_EVENT_DOWNLOAD_CHANNEL_STATUS

6.3.1.1.1.2 Context Errors

CRRC1: The ProactiveHandler and its content are not available for any toolkit applet triggered from the invocation to the termination of their processToolkit method for the following events:

EVENT_FIRST_COMMAND_AFTER_SELECT

6.3.1.1.2 Test Suite Files

Test Script:	FWK_MHA_PAHD_1.scr
Test Applet:	~~FWK_MHA_PAHD_1.java~~
	~~FWK_MHA_PAHD_2.java~~
~~Load Script:~~	~~FWK_MHA_PAHD_1.ldr~~
~~Cleanup Script:~~	~~FWK_MHA_PAHD_1.clr~~
~~Parameter File:~~	FWK_MHA_PAHD_1.java
	FWK_MHA_PAHD_2.java
Load Script:	FWK_MHA_PAHD_1.ldr
Cleanup Script:	FWK_MHA_PAHD_1.clr
Parameter File:	FWK_MHA_PAHD_1.par

6.3.1.1.3 Test Procedure

Id	Description	API /Framework Expectation	APDU Expectation
~~1~~	**~~Applets registration to all events and Proactive Handler availability with EVENT_PROFILE_DOWNLOAD~~** ~~Applet1 is registered to all events defined in [7].~~ ~~Using the methods initMenuEntry () for EVENT_MENU_SELECTION, requestPollInterval () for EVENT_STATUS_COMMAND, allocateTimer () for EVENT_TIMER_EXPIRATION and setEventList () for the rest of the events.~~ ~~Applet2 is registered to all events defined in [7], except EVENT_CALL_CONTROL_BY_SIM and EVENT_MO_SHORT_MESSAGE_CONTROL_BY_SIM.~~ ~~Using the methods initMenuEntry () for EVENT_MENU_SELECTION, requestPollInterval () for EVENT_STATUS_COMMAND, allocateTimer () for EVENT_TIMER_EXPIRATION and setEventList () for the rest of the events.~~ ~~The priority of applet1 is higher than priority of applet2~~ ~~1-Terminal Profile command is sent to SIM~~	~~1- Applet1 is triggered~~ ~~2- No exception is thrown.~~ ~~3- Applet2 is triggered~~ ~~4- No exception is thrown~~	

Id	Description	API /Framework Expectation	APDU Expectation
	~~without the facility of SET_EVENT_LIST, POLL_INTERVAL,SET UP IDLE MODE TEXT and SET UP MENU.~~ ~~2-Applet1 gets the Proactive Handler Applet1 is deregistered to EVENT_PROFILE_DOWNLOAD~~ ~~3-Applet2 gets the Proactive Handler Applet2 is deregistered to EVENT_PROFILE_DOWNLOAD~~	~~Applet1 finalizes~~	

Id	Description	API /Framework Expectation	APDU Expectation

1	**Applets registration to all events and Proactive Handler availability with EVENT_FIRST_COMMAND_AFTER_SELECT**
	
	`Applet1 is registered to all events defined in TS 43.019 [7]. Using the methods initMenuEntry() for EVENT_MENU_SELECTION, requestPollInterval() for EVENT_STATUS_COMMAND, allocateTimer() for EVENT_TIMER_EXPIRATION and setEventList() for the rest of the events.`
	
	`Applet2 is registered to all events defined in TS 43.019 [7], EVENT_CALL_CONTROL_BY_SIM and EVENT_MO_SHORT_MESSAGE_CONTROL_BY_SIM. Using the methods initMenuEntry() for EVENT_MENU_SELECTION, requestPollInterval() for EVENT_STATUS_COMMAND, allocateTimer() for EVENT_TIMER_EXPIRATION and setEventList() for the rest of the events.`
	
	`The priority of applet1 is higher than priority of applet2`
	`1- Select MF`
	
	`2- Applet1 gets the Proactive Handler. Applet1 is deregistered from EVENT_FIRST_COMMAND_AFTER_SELECT.`
	
	`3- Applet2 gets the Proactive Handler Applet2 is deregistered to EVENT_FIRST_COMMAND_AFTER_SELECT.`
		1- Applet1 is triggered by EVENT_FIRST_COMMAND_AFTER_SELECT
		
		2- A Toolkit Exception HANDLER_NOT_AVAILABLE is thrown.
		
		Applet1 finalizes
		
		Applet2 is triggered by EVENT_FIRST_COMMAND_AFTER_SELECT
		
		3- A Toolkit Exception HANDLER_NOT_AVAILABLE is thrown.
		Applet2 finalizes	

~~2~~	~~**Proactive Handler availability with EVENT_MENU_SELECTION_HELP_REQUEST**~~ ~~`Perform SIM initialization with all the facilities supported`~~ ~~`1-Envelope menu selection with help request is sent to the SIM`~~ ~~`2-Applet1 gets the Proactive Handler`~~ ~~`3-Envelope menu selection with help request is sent to the SIM`~~ ~~`4-Applet2 gets the Proactive Handler`~~	~~1- Applet1 is triggered~~ ~~2- No exception is thrown~~ ~~Applet1 finalizes~~ ~~3- Applet2 is triggered~~ ~~4- No exception is thrown~~	
2	**Proactive Handler availability with EVENT_PROFILE_DOWNLOAD** `1- Terminal Profile command is sent to the SIM without the facility of SET_EVENT_LIST, POLL_INTERVAL,SET UP IDLE MODE TEXT and SET UP MENU.`	1- Applet1 is triggered by EVENT_PROFILE_DOWNLOAD 2- No exception is thrown.	

Id	Description	API /Framework Expectation	APDU Expectation
	2- Applet1 gets the Proactive Handler Applet1 is deregistered to EVENT_PROFILE_DOWNLOAD 3- Applet2 gets the Proactive Handler Applet2 is deregistered to EVENT_PROFILE_DOWNLOAD	Applet1 finalizes. Applet2 is triggered by EVENT_PROFILE_DOWNLOAD 3- No exception is thrown	
~~3~~	**~~Proactive Handler availability with EVENT_MENU_SELECTION~~** ~~1-Envelope menu selection is sent to the SIM~~ ~~2-Applet1 gets the Proactive Handler~~ ~~3-Envelope menu selection is sent to the SIM~~ ~~4-Applet2 gets the Proactive Handler~~	~~1- Applet1 is triggered~~ ~~2- No exception is thrown.~~ ~~Applet1 finalizes~~ ~~3- Applet2 is triggered~~ ~~4- No exception is thrown.~~	
3	**Proactive Handler availability with EVENT_MENU_SELECTION_HELP_REQUEST** Perform SIM initialization with all the facilities supported 1- Envelope menu selection with help request is sent to the SIM 2- Applet1 gets the Proactive Handler	1- Applet1 is triggered 2- No exception is thrown Applet1 finalizes	
~~4~~	**~~Proactive Handler availability with EVENT_FORMATTED_SMS_PP_ENV~~** ~~1-Envelope dataDownLoad formatted is sent to the SIM~~ ~~2-Applet1 gets the Proactive Handler~~ ~~3-Envelope dataDownLoad formatted is sent to the SIM~~ ~~4-Applet2 gets the Proactive Handler~~	~~1- Applet1 is triggered~~ ~~2- No exception is thrown.~~ ~~Applet1 finalizes~~ ~~3- Applet2 is triggered~~ ~~4- No exception is thrown.~~	
4	**Proactive Handler availability with EVENT_MENU_SELECTION** 1- Envelope menu selection is sent to the SIM 2- Applet1 gets the Proactive Handler	1- Applet1 is triggered 2- No exception is thrown. Applet1 finalizes	
~~5~~	**~~Proactive Handler availability with~~**		

Id	Description	API /Framework Expectation	APDU Expectation
	~~EVENT_UNFORMATTED_SMS_PP_ENV~~ ~~1-Envelope dataDownLoad unformatted is sent to the SIM~~ ~~2-Applet1 gets the Proactive Handler~~ ~~3-Applet2 gets the Proactive Handler~~	~~1- Applet1 is triggered~~ ~~2- No exception is thrown.~~ ~~Applet1 finalizes~~ ~~3- Applet2 is triggered~~ ~~4- No exception is thrown.~~	

5	**Proactive Handler availability with EVENT_FORMATTED_SMS_PP_ENV**
	
	`1- Envelope dataDownLoad formatted is sent to the SIM`
	
	`2- Applet1 gets the Proactive Handler`
		1- Applet1 is triggered
		
		2- No exception is thrown.
		
		Applet1 finalizes	

6	**Proactive Handler availability with EVENT_UNFORMATTED_SMS_PP_ENV**
	
	`1- Envelope dataDownLoad unformatted is sent to the SIM`
	
	`2- Applet1 gets the Proactive Handler`
	
	`3- Applet2 gets the Proactive Handler`
		1- Applet1 is triggered
		
		2- No exception is thrown.
		
		Applet1 finalizes
		Applet2 is triggered
		
		3 No exception is thrown.	

~~6~~	~~**Proactive Handler availability with EVENT_FORMATTED_CELL BROADCAST**~~ ~~1-Envelope cell broadcast formatted is sent to the SIM~~ ~~2-Applet1 gets the Proactive Handler~~ ~~3-Envelope cell broadcast formatted is sent to the SIM~~ ~~4-Applet2 gets the Proactive Handler~~	~~1- Applet1 is triggered~~ ~~2-No exception is thrown~~ ~~Applet1 finalizes~~ ~~3-Applet2 is triggered~~ ~~4-No exception is thrown~~	
~~7~~	~~**Proactive Handler availability with EVENT_UNFORMATTED_CELL BROADCAST**~~ ~~1-Envelope cell broadcast unformatted is sent to the SIM~~ ~~2-Applet1 gets the Proactive Handler~~ ~~3-Applet2 gets the Proactive Handler~~	~~1- Applet1 is triggered~~ ~~2- No exception is thrown~~ ~~Applet1 finalizes~~ ~~3- Applet2 is triggered~~ ~~4- No exception is thrown~~	

Id	Description	API /Framework Expectation	APDU Expectation
7	**Proactive Handler availability with EVENT_FORMATTED_CELL_BROADCAST** 1- Envelope cell broadcast formatted is sent to the SIM 2- Applet1 gets the Proactive Handler	1- Applet1 is triggered 2-No exception is thrown Applet1 finalizes	
~~8~~	~~**Proactive Handler availability with EVENT_TIMER_EXPIRATION**~~ ~~1-Timer Id =1 Envelope Timer Expiration is sent to the SIM~~ ~~2-Applet1 gets the Proactive Handler~~ ~~3-Timer id=2 Envelope Timer Expiration is sent to the SIM~~ ~~4- Applet2 gets the Proactive Handler~~	~~1- Applet1 is triggered~~ ~~2- No exception is thrown.~~ ~~Applet1 finalizes~~ ~~3- Applet2 is triggered~~ ~~4- No exception is thrown~~	
8	**Proactive Handler availability with EVENT_UNFORMATTED_CELL_BROADCAST** 1- Envelope cell broadcast unformatted is sent to the SIM 2- Applet1 gets the Proactive Handler 3- Applet2 gets the Proactive Handler	1- Applet1 is triggered 2- No exception is thrown Applet1 finalizes Applet2 is triggered 3 No exception is thrown	
~~9~~	~~**Proactive Handler availability with EVENT_CALL_CONTROL_BY_SIM**~~ ~~1-Envelope call control by SIM is sent to the SIM~~ ~~2-Applet1 gets the Proactive Handler~~	~~1- Applet1 is triggered~~ ~~2- No exception is thrown.~~	
9	**Proactive Handler availability with EVENT_TIMER_EXPIRATION** 1- Timer Id =1 Envelope Timer Expiration is sent to the SIM 2- Applet1 gets the Proactive Handler	1- Applet1 is triggered 2- No exception is thrown. Applet1 finalizes	
10	**Proactive Handler availability with EVENT_CALL_CONTROL_BY_SIM** 1- Envelope call control by SIM is sent to the SIM 2- Applet1 gets the Proactive Handler	1- Applet1 is triggered 2- No exception is thrown.	
~~10~~	~~**Proactive Handler availability with EVENT_MO_SHORT_MESSAGE_CONTROL**~~	~~1- Applet1 is triggered~~	

Id	Description	API /Framework Expectation	APDU Expectation
	~~1-Envelope mo short message control by SIM is sent to the SIM~~ ~~2-Applet1 gets the Proactive Handler~~	~~2- No exception is thrown~~	
11	**Proactive Handler availability with EVENT_MO_SHORT_MESSAGE_CONTROL** 1- Envelope mo short message control by SIM is sent to the SIM 2- Applet1 gets the Proactive Handler	1- Applet1 is triggered 2- No exception is thrown	
~~11~~	~~**Proactive Handler availability with EVENT_EVENT_DOWNLOAD_MT_CALL**~~ ~~1-Envelope event download mt call is sent to the SIM~~ ~~2-Applet1 gets the Proactive Handler~~ ~~3-Applet2 gets the Proactive Handler~~	~~1- Applet1 is triggered~~ ~~2- No exception is thrown.~~ ~~Applet1 finalizes~~ ~~3- Applet2 is triggered~~ ~~4- No exception is thrown~~	
12	**Proactive Handler availability with EVENT_EVENT_DOWNLOAD_MT_CALL** 1- Envelope event download mt call is sent to the SIM 2- Applet1 gets the Proactive Handler 3- Applet2 gets the Proactive Handler	1- Applet1 is triggered 2- No exception is thrown. Applet1 finalizes Applet2 is triggered 3-No exception is thrown	
~~12~~	~~**Proactive Handler availability with EVENT_EVENT_DOWNLOAD_CALL_CONNECTED**~~ ~~1-Envelope event download call connected is sent to the SIM~~ ~~2-Applet1 gets the Proactive Handler~~ ~~3-Applet2 gets the Proactive Handler~~	~~1- Applet1 is triggered~~ ~~2- No exception is thrown.~~ ~~Applet1 finalizes~~ ~~3- Applet2 is triggered~~ ~~4- No exception is thrown~~	

Id	Description	API /Framework Expectation	APDU Expectation
13	**Proactive Handler availability with EVENT_EVENT_DOWNLOAD_CALL_CONNECTED** 1- Envelope event download call connected is sent to the SIM 2- Applet1 gets the Proactive Handler 3- Applet2 gets the Proactive Handler	1- Applet1 is triggered 2- No exception is thrown. Applet1 finalizes Applet2 is triggered 3- No exception is thrown	
~~13~~	~~**Proactive Handler availability with EVENT_EVENT_DOWNLOAD_CALL_DISCONNECTED**~~ ~~1-Envelope event download call disconnected is sent to the SIM~~ ~~2-Applet1 gets the Proactive Handler~~ ~~3-Applet2 gets the Proactive Handler~~	~~1- Applet1 is triggered~~ ~~2- No exception is thrown.~~ ~~Applet1 finalizes~~ ~~3- Applet2 is triggered~~ ~~4- No exception is thrown.~~	
14	**Proactive Handler availability with EVENT_EVENT_DOWNLOAD_CALL_DISCONNECTED** 1- Envelope event download call disconnected is sent to the SIM 2- Applet1 gets the Proactive Handler 3- Applet2 gets the Proactive Handler	1- Applet1 is triggered 2- No exception is thrown. Applet1 finalizes Applet2 is triggered 3- No exception is thrown.	
~~14~~	~~**Applets triggering with EVENT_EVENT_LOCATION_STATUS**~~ ~~1-Envelope event download location status is sent to the SIM~~ ~~2-Applet1 gets the Proactive Handler~~ ~~3-Applet2 gets the Proactive Handler~~	~~1- Applet1 is triggered~~ ~~2- No exception is thrown.~~ ~~Applet1 finalizes~~ ~~3- Applet2 is triggered~~ ~~4- No exception is thrown~~	

Id	Description	API /Framework Expectation	APDU Expectation

15	**Applets triggering with EVENT_EVENT_LOCATION_STATUS**
	
	`1- Envelope event download location status is sent to the SIM`
	
	`2- Applet1 gets the Proactive Handler`
	
	`3- Applet2 gets the Proactive Handler`
		1- Applet1 is triggered
		
		2- No exception is thrown.
		
		Applet1 finalizes
		Applet2 is triggered
		
		3- No exception is thrown	

~~15~~	~~**Proactive Handler availability with EVENT_EVENT_DOWNLOAD_USER_ACTIVITY**~~ ~~`1-Envelope event download user activity is sent to SIM`~~ ~~`2-Applet1 gets the Proactive Handler`~~ ~~`3-Applet2 gets the Proactive Handler`~~	~~1- Applet1 is triggered~~ ~~2- No exception is thrown~~ ~~Applet1 finalizes~~ ~~3- Applet2 is triggered~~ ~~4- No exception is thrown~~	

16	**Proactive Handler availability with EVENT_EVENT_DOWNLOAD_USER_ACTIVITY**
	
	`1- Envelope event download user activity is sent to the SIM`
	
	`2- Applet1 gets the Proactive Handler`
	
	`3- Applet2 gets the Proactive Handler`
		1- Applet1 is triggered
		
		2- No exception is thrown
		
		Applet1 finalizes
		Applet2 is triggered
		
		3- No exception is thrown	

~~16~~	~~**Proactive Handler availability with EVENT_EVENT_DOWNLOAD_IDLE_SCREEN_AVAILABLE**~~ ~~`1-Envelope event download idle screen available is sent to the SIM`~~ ~~`2-Applet1 gets the Proactive Handler`~~ ~~`3- Applet2 gets the Proactive Handler`~~	~~1- Applet1 is triggered~~ ~~2- No exception is thrown.~~ ~~Applet1 finalizes~~ ~~3- Applet2 is triggered~~ ~~4- No exception is thrown~~	

Id	Description	API /Framework Expectation	APDU Expectation

17	**Proactive Handler availability with EVENT_EVENT_DOWNLOAD_IDLE_SCREEN_AVAILABLE**
	`1- Envelope event download idle screen available is sent to the SIM`
	`2- Applet1 gets the Proactive Handler`
	`3- Applet2 gets the Proactive Handler`
		1- Applet1 is triggered
		2- No exception is thrown.
		Applet1 finalizes
		Applet2 is triggered
		3- No exception is thrown	

~~17~~	~~**Proactive Handler availability with EVENT_EVENT_DOWNLOAD_CARD_READER_STATUS**~~ ~~`1-Envelope event download card reader status is sent to the SIM`~~ ~~`2-Applet1 gets the Proactive Handler`~~ ~~`3-Applet2 gets the Proactive Handler`~~	~~1- Applet1 is triggered~~ ~~2- No exception is thrown.~~ ~~Applet1 finalizes~~ ~~3- Applet2 is triggered~~ ~~4- No exception is thrown~~	
~~18~~	~~**Proactive Handler availability with EVENT_EVENT_DOWNLOAD_LANGUAGE_SELECTION**~~ ~~`1-Envelope event download language selection is sent to the SIM`~~ ~~`2-Applet1 gets the Proactive Handler`~~ ~~`3-Applet2 gets the Proactive Handler`~~	~~1- Applet1 is triggered~~ ~~2-No exception is thrown.~~ ~~Applet1 finalizes~~ ~~Applet2 is triggered~~ ~~3-No exception is thrown~~	

Id	Description	API /Framework Expectation	APDU Expectation
18	**Proactive Handler availability with EVENT_EVENT_DOWNLOAD_CARD_READER_STATUS** 1- Envelope event download card reader status is sent to the SIM 2- Applet1 gets the Proactive Handler 3- Applet2 gets the Proactive Handler	1- Applet1 is triggered 2- No exception is thrown. Applet1 finalizes Applet2 is triggered 3- No exception is thrown	
19	**Proactive Handler availability with EVENT_EVENT_DOWNLOAD_LANGUAGE_SELECTION** 1- Envelope event download language selection is sent to the SIM 2- Applet1 gets the Proactive Handler 3- Applet2 gets the Proactive Handler	1- Applet1 is triggered 2-No exception is thrown. Applet1 finalizes Applet2 is triggered 3-No exception is thrown	
~~19~~	~~**Proactive Handler availability with EVENT_EVENT_DOWNLOAD_BROWSER_TERMINATION**~~ ~~1-Envelope event download browser termination is sent to the SIM~~ ~~2-Applet1 gets the Proactive Handler~~ ~~3-Applet2 gets the Proactive Handler~~	~~1- Applet1 is triggered~~ ~~2-No exception is thrown.~~ ~~Applet1 finalizes~~ ~~Applet2 is triggered~~ ~~3-No exception is thrown~~	
20	**Proactive Handler availability with EVENT_EVENT_DOWNLOAD_BROWSER_TERMINATION** 1- Envelope event download browser termination is sent to the SIM 2- Applet1 gets the Proactive Handler 3- Applet2 gets the Proactive Handler	1- Applet1 is triggered 2-No exception is thrown. Applet1 finalizes Applet2 is triggered 3-No exception is thrown	
~~20~~	~~**Proactive Handler availability with EVENT_STATUS_COMMAND**~~ ~~1-Status command is sent to the SIM~~ ~~2-Applet1 gets the Proactive Handler~~ ~~3- Applet2 gets the Proactive Handler~~	~~1- Applet1 is triggered~~ ~~2- No exception is thrown.~~ ~~Applet1 finalizes~~ ~~3- Applet2 is triggered~~ ~~4- No exception is thrown.~~	

Id	Description	API /Framework Expectation	APDU Expectation

21	**Proactive Handler availability with EVENT_STATUS_COMMAND**
	
	`1- Status command is sent to the SIM`
	
	`2- Applet1 gets the Proactive Handler`
	
	`3- Applet2 gets the Proactive Handler`
		1- Applet1 is triggered
		
		2- No exception is thrown.
		
		Applet1 finalizes
		
		Applet2 is triggered
		
		3- No exception is thrown.	

~~21~~	~~**Proactive Handler availability with UNRECOGNIZED_ENVELOPE**~~ ~~`1-An unrecognized Envelope (BER TLV Tag unrecognized) is sent to the SIM`~~ ~~`2-Applet1 gets the Proactive Handler`~~ ~~`3-Applet2 gets the Proactive Handler`~~	~~1- Applet1 is triggered~~ ~~2- No exception is thrown.~~ ~~Applet1 finalizes~~ ~~3- Applet2 is triggered~~ ~~4- No exception is thrown~~	

Id	Description	API /Framework Expectation	APDU Expectation
22	**Proactive Handler availability with EVENT_EVENT_DOWNLOAD_DATA_AVAILABLE** 1- Applet1 builds a proactive command OPEN CHANNEL proactiveHandler.send() method is called. 2- An Envelope Event Download Data Available is sent to the SIM, with channelId=01. 3- Applet1 gets the Proactive Handler	2-Applet1 is triggered 3-No exception is thrown. Applet1 finalizes	1- OPEN CHANNEL proactive Command is fetched TERMINAL RESPONSE is issued with Channel Id = 01
23	**Proactive Handler availability with EVENT_EVENT_DOWNLOAD_CHANNEL_STATUS** 1- An Envelope Event Download Channel Status is sent to the SIM, with ChannelId=01 2- Applet1 gets the Proactive Handler	1- Applet1 is triggered 2- No exception is thrown. Applet1 finalizes	
24	**Proactive Handler availability with UNRECOGNIZED_ENVELOPE** 1- An unrecognized Envelope (BER TLV Tag unrecognized) is sent to the SIM 2- Applet1 gets the Proactive Handler 3-Applet2 gets the Proactive Handler	1- Applet1 is triggered 2- No exception is thrown. Applet1 finalizes Applet2 is triggered 3- No exception is thrown	
25	**Proactive Handler availability with EVENT_FORMATTED_SMS_PP_UPD** 1- Update Record EFsms instruction formatted is sent to the SIM 2- Applet1 gets the Proactive Handler	1- Applet1 is triggered 2- No exception is thrown. Applet1 finalizes	
26	**Proactive Handler availability with EVENT_UNFORMATTED_SMS_PP_UPD** 1- Update Record EFsms instruction unformatted is sent to the SIM 2- Applet1 gets the Proactive Handler 3- Applet2 gets the Proactive Handler	1- Applet1 is triggered 2- No exception is thrown. Applet1 finalizes 3- Applet2 is triggered 4- No exception is thrown.	

~~6.3.1.1.4 Test Coverage~~

~~CRR Number~~	~~Test Case Number~~
~~CRRN1~~	~~1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 16, 17, 18, 19, 20, 21~~

6.3.1.1.4 Test Coverage

CRR Number	Test Case Number
CRRN1	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26
CRRC1	1

6.3.1.2 ProactiveResponseHandler

Test Area Reference: FWK_MHA_PRHD~~.~~

6.3.1.2.1 Conformance Requirement

6.3.1.2.1.1 Normal Execution

~~-~~CRRN1: The ProactiveResponseHandler is available after the first call to the ProactiveHandler.send() method to the termination of the processToolkit method for the following events:

EVENT_FORMATTED_SMS_PP_ENV

EVENT_UNFORMATTED_SMS_PP_ENV

EVENT_FORMATTED_SMS_PP_UPD

EVENT_UNFORMATTED_SMS_PP_UPD

EVENT_FORMATTED_SMS_CB

EVENT_UNFORMATTED_SMS_CB

EVENT_MENU_SELECTION

EVENT_MENU_SELECTION_HELP_REQUEST

EVENT_TIMER_EXPIRATION

EVENT_EVENT_DOWNLOAD_MT_CALL

EVENT_EVENT_DOWNLOAD_CALL_CONNECTED

EVENT_EVENT_DOWNLOAD_CALL_DISCONNECTED

EVENT_EVENT_DOWNLOAD_LOCATION_STATUS

EVENT_EVENT_DOWNLOAD_USER_ACTIVITY

EVENT_EVENT_DOWNLOAD_IDLE_SCREEN_AVAILABLE

EVENT_EVENT_DOWNLOAD_CARD_READER_STATUS

EVENT_EVENT_DOWNLOAD_LANGUAGE_SELECTION

EVENT_EVENT_DOWNLOAD_BROWSER_TERMINATION

EVENT_UNRECOGNIZED_ENVELOPE

EVENT_STATUS_COMMAND

EVENT_CALL_CONTROL

EVENT_SMS_MO_CONTROL

EVENT_PROFILE_DOWNLOAD

~~6.3.1.2.1.2 Parameters error~~

~~No requirements.~~

~~6.3.1.2.1.3 Context errors~~

~~No requirements.~~EVENT_EVENT_DOWNLOAD_DATA_AVAILABLE

EVENT_EVENT_DOWNLOAD_CHANNEL_STATUS

6.3.1.2.1.2 Context Errors

CRRC1: The ProactiveResponseHandler and its content are not available for any toolkit applet triggered from the invocation to the termination of their processToolkit method for the following events:

EVENT_FIRST_COMMAND_AFTER_SELECT

6.3.1.2.2 Test Suite Files

~~Test Script: FWK_MHA_PRHD_1.scr~~

~~Test Applet: FWK_MHA_PRHD_1.java~~

~~FWK_MHA_PRHD_2.java~~

~~Load Script: FWK_MHA_PRHD_1.ldr~~

~~Cleanup Script: FWK_MHA_PRHD_1.clr~~

~~Parameter File:~~

Test Script:	FWK_MHA_PRHD_1.scr
Test Applet:	FWK_MHA_PRHD_1.java
	FWK_MHA_PRHD_2.java
Load Script:	FWK_MHA_PRHD_1.ldr
Cleanup Script:	FWK_MHA_PRHD_1.clr
Parameter File:	FWK_MHA_PRHD_1.par

6.3.1.2.3 Test Procedure

Id	Description	API/Framework Expectation	APDU Expectation
~~1~~	**~~Applets registration to all events and Proactive Response Handler availability with EVENT_PROFILE_DOWNLOAD~~** ~~1- Applet1 is registered to all events defined in [7], applet2 is registered to all events defined in [7] except EVENT_CALL_CONTROL_BY_SIM and EVENT_MO_SMS_CONTROL_BY_SIM.~~ ~~Using the methods initMenuEntry() for EVENT_MENU_SELECTION, requestPollInterval() for EVENT_STATUS_COMMAND, allocateTimer() for EVENT_TIMER_EXPIRATION and setEventList() for the rest of the events.~~ ~~1-Terminal Profile command is sent to the SIM without the facility of SET_EVENT_LIST and POLL_INTERVAL, ,SET UP IDLE MODE TEXT and SET UP MENU.~~ ~~Applet1 builds a proactive command DISPLAY TEXT.~~ ~~2- ProactiveHandler.send() method is called~~ ~~3- ProactiveResponseHandler.getTheHandler() method is called~~ ~~Applet1 is deregistered to EVENT_PROFILE_DOWNLOAD~~ ~~Applet2 builds a proactive command DISPLAY TEXT.~~ ~~4- ProactiveHandler.send() method is called~~ ~~5- ProactiveResponseHandler.getTheHandler() method is called~~ ~~Applet1 is deregistered to EVENT_PROFILE_DOWNLOAD~~	~~1-Applet1 is triggered~~ ~~No exception is thrown~~ ~~3- No exception is thrown~~ ~~Applet1 finalizes~~ ~~4- Applet2 is triggered~~ ~~6- No exception is thrown~~	~~2- The proactive command DISPLAY TEXT is fetched~~ ~~TERMINAL RESPONSE~~ ~~5- The proactive command DISPLAY TEXT is fetched~~ ~~TERMINAL RESPONSE~~
1	**Applets registration to all events and Proactive Response Handler availability with EVENT_PROFILE_DOWNLOAD** Applet1 is registered to all events defined in TS 43.019 [7] except EVENT_FIRST_COMMAND_AFTER_SELECT, Applet2 is registered to all events defined in TS 43.109~~-~~[7] except EVENT_FIRST_COMMAND_AFTER_SELECT, EVENT_CALL_CONTROL_BY_SIM and EVENT_MO_SMS_CONTROL_BY_SIM. Using the methods initMenuEntry() for EVENT_MENU_SELECTION, requestPollInterval() for EVENT_STATUS_COMMAND, allocateTimer() for EVENT_TIMER_EXPIRATION and setEventList() for the rest of the events.		

Id	Description	API/Framework Expectation	APDU Expectation
	1-Terminal Profile command is sent to the SIM without the facility of SET_EVENT_LIST, POLL_INTERVAL,SET UP IDLE MODE TEXT and SET UP MENU.	1-Applet1 is triggered by EVENT_PROFILE_DOWNLOAD No exception is thrown	
	2- Applet1 builds a proactive command DISPLAY TEXT.		
	3- ProactiveHandler.send() method is called		3- The proactive command DISPLAY TEXT is fetched
	4- ProactiveResponseHandler.getTheHandler() method is called	4- No exception is thrown	TERMINAL RESPONSE
	Applet1 is deregistered to EVENT_PROFILE_DOWNLOAD		
	5- Applet2 builds a proactive command DISPLAY TEXT.	Applet1 finalizes	
	6- ProactiveHandler.send() method is called	Applet2 is triggered by EVENT_PROFILE_DOWNLOAD	
	7- ProactiveResponseHandler.getTheHandler() method is called	7- No exception is thrown	6- The proactive command DISPLAY TEXT is fetched TERMINAL RESPONSE
	Applet1 is deregistered to EVENT_PROFILE_DOWNLOAD		

Id	Description	API/Framework Expectation	APDU Expectation
~~2~~	~~**Proactive Response Handler availability with EVENT_MENU_SELECTION_HELP_REQUEST**~~		

2	**Proactive Response Handler availability with EVENT_MENU_SELECTION_HELP_REQUEST**
	
	`Perform SIM initialization with all the facilities supported`
	
	`1-Envelope menu selection with help request is sent to the 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

~~3~~	~~**Proactive Response Handler availability with EVENT_MENU_SELECTION**~~ ~~`1-Envelope menu selection is sent to the 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~~ ~~Applet1 finalizes~~	~~2- A proactive command DISPLAY TEXT is fetched~~ ~~TERMINAL RESPONSE~~

Id	Description	API/Framework Expectation	APDU Expectation
	~~4-Envelope menu selection is sent to the SIM~~ ~~Applet2 builds a proactive command DISPLAY TEXT~~ ~~5- ProactiveHandler.send() method is called~~ ~~6-ProactiveResponseHandler.getTheHandler() method is called~~	~~4- Applet2 is triggered~~ ~~6- No exception is thrown~~	~~5- A proactive command DISPLAY TEXT is fetched~~ ~~TERMINAL RESPONSE~~
3	**Proactive Response Handler availability with EVENT_MENU_SELECTION** 1-Envelope menu selection is sent to the 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

Id	Description	API/Framework Expectation	APDU Expectation
~~4~~	~~**Proactive Response Handler availability with EVENT_FORMATTED_SMS_PP_ENV**~~ ~~1-Envelope dataDownLoad formatted is sent to the SIM~~ ~~Applet builds a proactive command DISPLAY TEXT~~ ~~2-ProactiveHandler.send() method is called~~ ~~3-ProactiveResponseHandler.getTheHandler() method is called~~ ~~4-Envelope dataDownLoad formatted is sent to the SIM~~ ~~Applet2 builds a proactive command DISPLAY TEXT~~ ~~5-ProactiveHandler.send() method is called~~ ~~6-ProactiveResponseHandler.getTheHandler() method is called~~	~~1- Applet1 is triggered~~ ~~3- No exception is thrown~~ ~~Applet1 finalizes~~ ~~4- Applet2 is triggered~~ ~~6- No exception is thrown~~	~~2- A proactive command DISPLAY TEXT is fetched~~ ~~TERMINAL RESPONSE~~ ~~5- A proactive command DISPLAY TEXT is fetched~~ ~~TERMINAL RESPONSE~~
4	**Proactive Response Handler availability with EVENT_FORMATTED_SMS_PP_ENV** 1-Envelope dataDownLoad formatted is sent to the 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
~~5~~	~~**Proactive Response Handler availability with EVENT_UNFORMATTED_SMS_PP_ENV**~~ ~~1-Envelope dataDownLoad unformatted is sent to the 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~~	~~1- Applet1 is triggered~~ ~~3- No exception is thrown~~ ~~Applet1 finalizes~~ ~~4- Applet2 is triggered~~	~~2- A proactive command DISPLAY TEXT is fetched~~ ~~TERMINAL RESPONSE~~ ~~5- A proactive command~~

Id	Description	API/Framework Expectation	APDU Expectation
	~~called~~ ~~5- ProactiveResponseHandler.getTheHandler() method is called~~	~~6- No exception is thrown~~	~~DISPLAY TEXT is fetched~~ ~~TERMINAL RESPONSE~~
5	**Proactive Response Handler availability with EVENT_UNFORMATTED_SMS_PP_ENV** 1-Envelope dataDownLoad unformatted is sent to the SIM Applet1 builds a proactive command DISPLAY TEXT 2- ProactiveHandler.send() method is called 3- ProactiveResponseHandler.getTheHandler() method is called Applet2 builds a proactive command DISPLAY TEXT 4- ProactiveHandler.send() method is called 5- ProactiveResponseHandler.getTheHandler() method is called	1- Applet1 is triggered 3- No exception is thrown Applet1 finalizes Applet2 is triggered 5- No exception is thrown	2- A proactive command DISPLAY TEXT is fetched TERMINAL RESPONSE 4- A proactive command DISPLAY TEXT is fetched TERMINAL RESPONSE
~~6~~	~~**Proactive Response Handler availability with EVENT_UNFORMATTED_SMS_CB**~~ ~~1-Envelope call broadcast unformatted is sent to the 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~~ ~~4- Applet2 is triggered~~ ~~6- No exception is thrown~~	~~2- A proactive command DISPLAY TEXT is fetched~~ ~~TERMINAL RESPONSE~~ ~~5- A proactive command DISPLAY TEXT is fetched~~ ~~TERMINAL RESPONSE~~
6	**Proactive Response Handler availability with EVENT_FORMATTED_SMS_CB**		

Id	Description	API/Framework Expectation	APDU Expectation
	1-Envelope cell broadcast formatted is sent to the 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
~~7~~	**~~Proactive Response Handler availability with EVENT_UNFORMATTED_SMS_CB~~** ~~1-Envelope call broadcast unformatted is sent to the 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~~ ~~4- Applet2 is triggered~~ ~~6- No exception is thrown~~	~~2- A proactive command DISPLAY TEXT is fetched~~ ~~TERMINAL RESPONSE~~ ~~5- A proactive command DISPLAY TEXT is fetched~~ ~~TERMINAL RESPONSE~~

Id	Description	API/Framework Expectation	APDU Expectation
7	**Proactive Response Handler availability with EVENT_UNFORMATTED_SMS _CB** 1-Envelope call broadcast unformatted is sent to the 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
~~8~~	~~**Proactive Response Handler availability with EVENT_TIMER_EXPIRATION**~~ ~~Timer id=1~~ ~~1-Envelope Timer Expiration is sent to the SIM~~ ~~Applet builds a proactive command DISPLAY TEXT~~ ~~2-ProactiveHandler.send() method is called~~ ~~3-ProactiveResponseHandler.getTheHandler() method is called~~ ~~Timer id=2~~ ~~Envelope Timer Expiration is sent to the SIM~~ ~~Applet 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~~ ~~4- Applet2 is triggered~~ ~~5- No exception is thrown~~	~~2- A proactive command DISPLAY TEXT is fetched~~ ~~TERMINAL RESPONSE~~ ~~6- A proactive command DISPLAY TEXT is fetched~~ ~~TERMINAL RESPONSE~~
8	**Proactive Response Handler availability with EVENT_TIMER_EXPIRATION**		

Id	Description	API/Framework Expectation	APDU Expectation
	Timer id=1 1-Envelope Timer Expiration is sent to the SIM Applet builds a proactive command DISPLAY TEXT 2-ProactiveHandler.send() method is called 3-ProactiveResponseHandler.getTheHandler() method is called	1- Applet1 is triggered 3- No exception is thrown	2- A proactive command DISPLAY TEXT is fetched TERMINAL RESPONSE
~~9~~	~~**Proactive Response Handler availability with EVENT_CALL_CONTROL_BY_SIM**~~ ~~1-Envelope call control by sim is sent to the 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/Framework Expectation	APDU Expectation
9	**Proactive Response Handler availability with EVENT_CALL_CONTROL_BY_SIM** 1-Envelope call control by sim is sent to the 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
~~10~~	~~**Proactive Response Handler availability with MO_SHORT_MESSAGE_CONTROL_BY_SIM**~~ ~~1-Envelope mo short message control by sim is sent to the 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~~
10	**Proactive Response Handler availability with MO_SHORT_MESSAGE_CONTROL_BY_SIM** 1-Envelope mo short message control by sim is sent to the 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
~~11~~	~~**Proactive Response Handler availability with EVENT_EVENT_DOWNLOAD_MT_CALL**~~ ~~1-Envelope event download mt call is sent to the 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~~ ~~Applet1 finalizes~~	~~2- A proactive command DISPLAY TEXT is fetched~~ ~~TERMINAL RESPONSE~~

Id	Description	API/Framework Expectation	APDU Expectation
	~~Applet2 builds a proactive command DISPLAY TEXT~~ ~~4- ProactiveHandler.send() method is called~~ ~~5- ProactiveResponseHandler.getTheHandler() method is called~~	~~4- Applet2 is triggered~~ ~~6- No exception is thrown~~	~~5- A proactive command DISPLAY TEXT is fetched~~ ~~TERMINAL RESPONSE~~

Id	Description	API/Framework Expectation	APDU Expectation
11	**Proactive Response Handler availability with EVENT_EVENT_DOWNLOAD_MT_CALL** 1-Envelope event download mt call is sent to the 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
~~12~~	~~**Proactive Response Handler availability with EVENT_EVENT_DOWNLOAD_CALL_CONNECTED**~~ ~~1-Envelope event download call connected is sent to the SIM~~ ~~Applet1 builds a proactive command DISPLAY TEXT~~ ~~2-ProactiveHandler.send() method is called~~ ~~3-ProactiveResponseHandler.getTheHandler() method is called~~ ~~Applet 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~~ ~~4- Applet2 is triggered~~ ~~6- No exception is thrown~~	~~2- A proactive command DISPLAY TEXT is fetched~~ ~~TERMINAL RESPONSE~~ ~~5- A proactive command DISPLAY TEXT is fetched~~ ~~TERMINAL RESPONSE~~
12	**Proactive Response Handler availability with EVENT_EVENT_DOWNLOAD_CALL_CONNECTED** 1-Envelope event download call connected is sent to the SIM Applet1 builds a proactive command DISPLAY TEXT 2-ProactiveHandler.send() method is called	1- Applet1 is triggered	2- A proactive command DISPLAY TEXT is fetched

Id	Description	API/Framework Expectation	APDU Expectation
	3-ProactiveResponseHandler.getTheHandler() method is called Applet builds a proactive command DISPLAY TEXT 4- ProactiveHandler.send() method is called 5- ProactiveResponseHandler.getTheHandler() method is called	3- No exception is thrown Applet1 finalizes Applet2 is triggered 5- No exception is thrown	TERMINAL RESPONSE 4- A proactive command DISPLAY TEXT is fetched TERMINAL RESPONSE
~~13~~	~~**Proactive Response Handler availability with EVENT_EVENT_DOWNLOAD_CALL_DISCONNECTED**~~ ~~1-Envelope event download call disconnected is sent to the 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~~ ~~4- Applet2 is triggered~~ ~~6- No exception is thrown~~	~~2- A proactive command DISPLAY TEXT is fetched~~ ~~TERMINAL RESPONSE~~ ~~5- A proactive command DISPLAY TEXT is fetched~~ ~~TERMINAL RESPONSE~~

Id	Description	API/Framework Expectation	APDU Expectation

13	**Proactive Response Handler availability with EVENT_EVENT_DOWNLOAD_CALL_DISCONNECTED**
	
	`1-Envelope event download call disconnected is sent to the 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

~~14~~	~~**Proactive Response Handler availability with EVENT_EVENT_DOWNLOAD_LOCATION_STATUS**~~ ~~`1-Envelope event download location status is sent to the 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~~ ~~4- Applet2 is triggered~~ ~~6- No exception is thrown~~	~~2-A proactive command DISPLAY TEXT is fetched~~ ~~TERMINAL RESPONSE~~ ~~5- A proactive command DISPLAY TEXT is fetched~~ ~~TERMINAL RESPONSE~~
14	**Proactive Response Handler availability with EVENT_EVENT_DOWNLOAD_LOCATION_STATUS** `1-Envelope event download location status`	1- Applet1 is triggered	

Id	Description	API/Framework Expectation	APDU Expectation
	is sent to the 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	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
~~15~~	~~**Proactive Response Handler availability with EVENT_EVENT_DOWNLOAD_USER_ACTIVITY**~~ ~~1-Envelope event download user activity is sent to the 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~~ ~~4- Applet2 is triggered~~ ~~6- No exception is thrown~~	~~2-A proactive command DISPLAY TEXT is fetched~~ ~~TERMINAL RESPONSE~~ ~~5- A proactive command DISPLAY TEXT is fetched~~ ~~TERMINAL RESPONSE~~

Id	Description	API/Framework Expectation	APDU Expectation
15	**Proactive Response Handler availability with EVENT_EVENT_DOWNLOAD_USER_ACTIVITY** 1-Envelope event download user activity is sent to the 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
~~16~~	~~**Proactive Response Handler availability with EVENT_EVENT_DOWNLOAD_IDLE_SCREEN_AVAILABLE**~~ ~~1-Envelope event download idle screen available is sent to the 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~~ ~~4- Applet2 is triggered~~ ~~6- No exception is thrown~~	~~2- A proactive command DISPLAY TEXT is fetched~~ ~~TERMINAL RESPONSE~~ ~~5- A proactive command DISPLAY TEXT is fetched~~ ~~TERMINAL RESPONSE~~
16	**Proactive Response Handler availability with EVENT_EVENT_DOWNLOAD_IDLE_SCREEN_AVAILABLE** 1-Envelope event download idle screen available is sent to the SIM Applet1 builds a proactive command DISPLAY TEXT	1- Applet1 is triggered	

Id	Description	API/Framework Expectation	APDU Expectation

	`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`
		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/Framework Expectation	APDU Expectation
~~17~~	~~**Proactive Response Handler availability with EVENT_EVENT_DOWNLOAD_LANGUAGE_**~~		

17	**Proactive Response Handler availability with EVENT_EVENT_DOWNLOAD_CARD_READER_STATUS**
	
	`1-Envelope event download card reader status is sent to the 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/Framework Expectation	APDU Expectation
~~18~~	~~**Proactive Response Handler availability with EVENT_EVENT_DOWNLOAD_BROWSER_**~~		

18	**Proactive Response Handler availability with EVENT_EVENT_DOWNLOAD_LANGUAGE_ SELECTION**
	
	`1-Envelope event download language selection is sent to the 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/Framework Expectation	APDU Expectation
~~19~~	~~**Proactive Response Handler availability with EVENT_STATUS_COMMAND**~~		
19	**Proactive Response Handler availability with EVENT_EVENT_DOWNLOAD_BROWSER_ TERMINATION**		
	1-Envelope event download Browser termination is sent to the SIM	1- Applet1 is triggered	
	Applet1 builds a proactive command DISPLAY TEXT		
	2-ProactiveHandler.send() method is called		2-A proactive command DISPLAY TEXT is fetched TERMINAL RESPONSE
	3-ProactiveResponseHandler.getTheHandler() method is called	3-No exception is thrown Applet1 finalizes Applet2 is triggered	
	Applet2 builds a proactive command DISPLAY TEXT		
	4- ProactiveHandler.send() method is called		4-A proactive command DISPLAY TEXT is fetched
	5-ProactiveResponseHandler.getTheHandler() method is called	5-No exception is thrown	TERMINAL RESPONSE
~~20~~	~~**Proactive Response Handler availability with UNRECOGNIZED_ENVELOPE**~~		
	~~1-An unrecognized Envelope is sent to the SIM~~	~~1- Applet1 is triggered~~	
	~~Applet1 builds a proactive command DISPLAY TEXT~~		

Id	Description	API/Framework Expectation	APDU Expectation
	~~2-ProactiveHandler.send() method is called~~		~~2- A proactive command DISPLAY TEXT is fetched~~
			~~TERMINAL RESPONSE~~
	~~3-ProactiveResponseHandler.getTheHandler() method is called~~	~~3- No exception is thrown~~	
		~~Applet1 finalizes~~	
	~~Applet2 builds a proactive command DISPLAY TEXT~~	~~4- Applet2 is triggered~~	~~5- A proactive command DISPLAY TEXT is fetched~~
	~~4- ProactiveHandler.send() method is called~~		~~TERMINAL RESPONSE~~
	~~5- ProactiveResponseHandler.getTheHandler() method is called~~	~~6- No exception is thrown~~	
20	**Proactive Response Handler availability with EVENT_STATUS_COMMAND**		
	1-Status command is sent to the SIM	1- Applet1 is triggered	
	Applet1 builds a proactive command DISPLAY TEXT		
	2-ProactiveHandler.send() method is called		2- A proactive command DISPLAY TEXT is fetched
			TERMINAL RESPONSE
	3-ProactiveResponseHandler.getTheHandler() method is called	3- No exception is thrown	
		Applet1 finalizes Applet2 is triggered	
	Applet2 builds a proactive command DISPLAY TEXT		
	4- ProactiveHandler.send() method is called		4- A proactive command DISPLAY TEXT is fetched
			TERMINAL RESPONSE
	5- ProactiveResponseHandler.getTheHandler() method is called	5- No exception is thrown	
21	**Proactive Handler availability with EVENT_EVENT_DOWNLOAD_DATA_AVAILABLE**		
	1- Applet1 builds a proactive command OPEN CHANNEL. proactiveHandler.send() method is called	1- Applet1 is registered to EVENT_EVENT_DOWNLOAD_DATA_AVAILABLE and EVENT_EVENT_DOWNLOAD_CHANNEL_STATUS	1- OPEN CHANNEL proactive command is fetched
			TERMINAL RESPONSE is issued with Channel Id = 01
	2- An Envelope Event Download Data Available is sent to the SIM, with ChannelId=01.	2- Applet1 is triggered	
	3-Applet1 builds a proactive command DISPLAY TEXT		
	4- ProactiveHandler.send() method is called		4- A proactive command DISPLAY TEXT is fetched

Id	Description	API/Framework Expectation	APDU Expectation
	5- ProactiveResponseHandler.getTheHandler() method is called	5- No exception is thrown	TERMINAL RESPONSE
22	**Proactive Handler availability with EVENT_EVENT_DOWNLOAD_CHANNEL_STATUS** 1-An Envelope Event Download Channel Status is sent to the SIM with ChannelId=01. 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

Id	Description	API/Framework Expectation	APDU Expectation
23	**Proactive Response Handler availability with UNRECOGNIZED_ENVELOPE** 1-An unrecognized Envelope is sent to the 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
24	**Proactive Response Handler availability with EVENT_FORMATTED_SMS_PP_UPD** 1- Update Record EFsms instruction formatted is sent to the 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
25	**Proactive Response Handler availability with EVENT_UNFORMATTED_SMS_PP_UPD** 1- Update Record EFsms instruction unformatted is sent to the 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-	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

Id	Description	API/Framework Expectation	APDU Expectation
	`ProactiveResponseHandler.getTheHandler() method is called`	5- No exception is thrown	

6.3.1.2.4 Test Coverage

CRR Number	Test Case Number
~~CRRN1~~	~~1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20~~
CRRN1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25
CRRC1	Not testable

6.3.1.3 EnvelopeHandler

Test Area Reference: FWK_MHA_ENHD~~.~~

6.3.1.3.1 Conformance Requirement

6.3.1.3.1.1 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_UNFORMATTED_SMS_PP_ENV

EVENT_FORMATTED_SMS_PP_UPD

EVENT_UNFORMATTED_SMS_PP_UPD

EVENT_FORMATTED_SMS_CB

EVENT_UNFORMATTED_SMS_CB

EVENT_MENU_SELECTION

EVENT_MENU_SELECTION_HELP_REQUEST

EVENT_TIMER_EXPIRATION

EVENT_EVENT_DOWNLOAD_MT_CALL

EVENT_EVENT_DOWNLOAD_CALL_CONNECTED

EVENT_EVENT_DOWNLOAD_CALL_DISCONNECTED

EVENT_EVENT_DOWNLOAD_LOCATION_STATUS

EVENT_EVENT_DOWNLOAD_USER_ACTIVITY

EVENT_EVENT_DOWNLOAD_IDLE_SCREEN_AVAILABLE

EVENT_EVENT_DOWNLOAD_CARD_READER_STATUS

EVENT_EVENT_DOWNLOAD_LANGUAGE_SELECTION

EVENT_EVENT_DOWNLOAD_BROWSER_TERMINATION

EVENT_UNRECOGNIZED_ENVELOPE

EVENT_CALL_CONTROL

EVENT_SMS_MO_CONTROL

~~6.3.1.3.1.2 Parameters error~~

~~No requirements.~~EVENT_EVENT_DOWNLOAD_DATA_AVAILABLE

EVENT_EVENT_DOWNLOAD_CHANNEL_STATUS

6.3.1.3.1.2 Context Errors

CRRC1: The EnvelopeHandler and its content are not available for any toolkit applet triggered from the invocation to the termination of their processToolkit method for the following events:

~~EVENT_STATUS_COMMAND~~

~~EVENT_PROFILE_DOWNLOAD~~ EVENT_STATUS_COMMAND

EVENT_PROFILE_DOWNLOAD

EVENT_FIRST_COMMAND_AFTER_SELECT

6.3.1.3.2 Test Suite Files

~~Test Script: FWK_MHA_ENHD_1.scr~~

~~Test Applet: FWK_MHA_ENHD_1.java~~

~~FWK_MHA_ENHD_2.java~~

~~Load Script: FWK_MHA_ENHD_1.ldr~~

~~Cleanup Script: FWK_MHA_ENHD_1.clr~~

~~Parameter File: FWK_MHA_ENHD_1.par~~

~~6.3.1.3.3 Test Procedure~~

Test Script:	FWK_MHA_ENHD_1.scr
Test Applet:	FWK_MHA_ENHD_1.java
	FWK_MHA_ENHD_2.java
Load Script:	FWK_MHA_ENHD_1.ldr
Cleanup Script:	FWK_MHA_ENHD_1.clr
Parameter File:	FWK_MHA_ENHD_1.par

6.3.1.3.3 Test Procedure

Id	Description	API/Framework Expectation	APDU Expectation
~~1~~	**~~Applet1 and Applet2 registration and Envelope Handler availability with EVENT_PROFILE_DOWNLOAD~~** ~~1- Applet1 is registered to all events defined [7].~~ ~~Using the methods initMenuEntry() for EVENT_MENU_SELECTION, requestPollInterval() for EVENT_STATUS_COMMAND, allocateTimer() for EVENT_TIMER_EXPIRATION and setEventList() for the rest of the events.~~ ~~Applet2 is registered to all events defined [7] except EVENT_CALL_CONTROL_BY_SIM and EVENT_MO_SHORT_MESSAGE_CONTROL_BY_SIM.~~ ~~Using the methods initMenuEntry() for EVENT_MENU_SELECTION, requestPollInterval() for EVENT_STATUS_COMMAND, allocateTimer for EVENT_TIMER_EXPIRATION and setEventList for the rest of the events.~~ ~~2-Terminal Profile command is sent to SIM without the facility of SET_EVENT_LIST ,SETUP_IDLE_MODE_TEXT ,POLL_INTERVAL and SETUP MENU~~ ~~3-EnvelopeHandler.getTheHandler() method is called by Applet1~~ ~~Applet1 is deregistered to EVENT_PROFILE_DOWNLOAD~~ ~~4-EnvelopeHandler.getTheHandler() method is called by Applet2~~ ~~Applet2 is deregistered to EVENT_PROFILE_DOWNLOAD~~	~~1- No exception is thrown~~ ~~2- Applet1 is triggered~~ ~~3- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown~~ ~~4- Applet2 is triggered~~ ~~5- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown~~	

Id	Description	API/Framework Expectation	APDU Expectation
~~2~~	~~**Envelope Handler availability with EVENT_MENU_SELECTION_HELP_REQUEST**~~		

Id	Description	API/Framework Expectation	APDU Expectation
1	**Applet1 and Applet2 registration and Envelope Handler availability with EVENT_FIRST_COMMAND_AFTER_SELECT** 1.Applet1 is registered to all events defined TS 43.019 [7]. The registration is done using the methods initMenuEntry() for EVENT_MENU_SELECTION, requestPollInterval() for EVENT_STATUS_COMMAND, allocateTimer() for EVENT_TIMER_EXPIRATION and setEventList() for the rest of the events. Applet2 is registered to all events defined TS 43.019 [7] except EVENT_PROFILE_DOWNLOAD, EVENT_CALL_CONTROL_BY_SIM and EVENT_MO_SHORT_MESSAGE_CONTROL_BY_SIM. The registration is done using the methods initMenuEntry() for EVENT_MENU_SELECTION, requestPollInterval() for EVENT_STATUS_COMMAND, allocateTimer for EVENT_TIMER_EXPIRATION and setEventList for the rest of the events. 2- Select MF. 3-EnvelopeHandler.getTheHandler() method is called by Applet1 Applet1 is deregistered from EVENT_FIRST_COMMAND_AFTER_SELECT. 4-EnvelopeHandler.getTheHandler() method is called by Applet2 Applet2 is deregistered to EVENT_FIRST_COMMAND_AFTER_SELECT.	1- No exception is thrown 2- Applet1 is triggered by EVENT_FIRST_COMMAND_AFTER_SELECT 3- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown Applet1 finalizes Applet2 is triggered 4- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown	
2	**Handler availability with EVENT_PROFILE_DOWNLOAD** 1- Terminal Profile command is sent to the SIM without the facility of SET_EVENT_LIST, SETUP_IDLE_MODE_TEXT, POLL_INTERVAL and SETUP MENU 2- EnvelopeHandler.getTheHandler() method is called by Applet1 Applet1 is deregistered to EVENT_PROFILE_DOWNLOAD 3-EnvelopeHandler.getTheHandler() method is called by Applet2 Applet2 is deregistered to EVENT_PROFILE_DOWNLOAD	1- Applet1 is triggered by EVENT_PROFILE_DOWNLOAD 2- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown Applet1 finalizes Applet2 is triggered by EVENT_PROFILE_DOWNLOAD 3- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown	
~~3~~	~~**Envelope Handler availability with EVENT_MENU_SELECTION**~~ ~~1-Envelope menu selection is sent to the SIM~~ ~~2-EnvelopeHandler.getTheHandler() method is called by Applet1~~ ~~3-Envelope menu selection is sent to the SIM~~	~~1-Applet1 is triggered~~ ~~2-No exception is thrown.~~ ~~Applet1 finalizes~~ ~~3-Applet2 is triggered~~	

Id	Description	API/Framework Expectation	APDU Expectation
	~~4-EnvelopeHandler.getTheHandler() method is called by Applet2~~	~~4- No exception is thrown.~~	
3	**Envelope Handler availability with EVENT_MENU_SELECTION_HELP_REQUEST** Perform SIM initialization with all the facilities supported Envelope menu selection with help request is sent to the SIM 1-EnvelopeHandler.getTheHandler() method is called by Applet1	1- Applet1 is triggered 2- No exception is thrown.	
4	~~**Envelope Handler availability with EVENT_FORMATTED_SMS_PP_ENV**~~ ~~1-A EVENT_FORMATTED_SMS_PP_ENV envelope is sent to the SIM~~ ~~2-EnvelopeHandler.getTheHandler() method is called by Applet1~~ ~~3-A EVENT_FORMATTED_SMS_PP_ENV envelope is sent to the SIM~~ ~~4-EnvelopeHandler.getTheHandler() method is called by Applet2~~	~~1- Applet1 is triggered~~ ~~2- No exception is thrown.~~ ~~Applet1 finalizes~~ ~~3- Applet2 is triggered~~ ~~4- No exception is thrown.~~	
4	**Envelope Handler availability with EVENT_MENU_SELECTION** 1-Envelope menu selection is sent to the SIM 2-EnvelopeHandler.getTheHandler() method is called by Applet1	1- Applet1 is triggered 2- No exception is thrown.	

Id	Description	API/Framework Expectation	APDU Expectation
~~5~~	~~**Envelope Handler availability with EVENT_UNFORMATTED_SMS_PP_ENV**~~		

5	**Envelope Handler availability with EVENT_FORMATTED_SMS_PP_ENV**
	`1-A EVENT_FORMATTED_SMS_PP_ENV envelope is sent to the SIM`
	`2-EnvelopeHandler.getTheHandler() method is called by Applet1`
		1- Applet1 is triggered
		2- No exception is thrown.	

6	**Envelope Handler availability with EVENT_UNFORMATTED_SMS_PP_ENV**
	`1-An unformatted sms pp envelope is sent to the SIM`
	`2-EnvelopeHandler.getTheHandler() method is called by Applet1`
	`3-EnvelopeHandler.getTheHandler() method is called by Applet2`
		1- Applet1 is triggered
		2- No exception is thrown.
		Applet1 finalizes
		3- Applet2 is triggered
		4- No exception is thrown.	

~~6~~	~~**Envelope Handler availability with EVENT_FORMATTED_CB**~~ ~~`1-Envelope cell broadcast formatted is sent to the SIM`~~ ~~`2-EnvelopeHandler.getTheHandler() method is called by Applet1`~~ ~~`3- Envelope cell broadcast formatted is sent to the SIM`~~ ~~`4-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/Framework Expectation	APDU Expectation

7	**Envelope Handler availability with EVENT_FORMATTED_CB**
	
	`1-Envelope cell broadcast formatted is sent to the SIM`
	
	`2-EnvelopeHandler.getTheHandler() method is called by Applet1`
		1- Applet1 is triggered
		
		2-No exception is thrown	

~~7~~	~~**Envelope Handler availability with EVENT_UNFORMATTED_CB**~~ ~~`1-Envelope cell broadcast unformatted is sent to the 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~~	

8	**Envelope Handler availability with EVENT_UNFORMATTED_CB**
	
	`1-Envelope cell broadcast unformatted is sent to the 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/Framework Expectation	APDU Expectation
~~8~~	~~**Envelope Handler availability with EVENT_TIMER_EXPIRATION**~~		
~~9~~	~~**Envelope Handler availability with EVENT_CALL_CONTROL_BY_SIM**~~ ~~`1-Envelope call control by sim is sent to the SIM`~~ ~~`2-EnvelopeHandler.getTheHandler() method is called by Applet1`~~	~~1- Applet1 is triggered~~ ~~2- No exception is thrown.~~	

9	**Envelope Handler availability with EVENT_TIMER_EXPIRATION**
	`Timer id=1`
	`1-Envelope Timer Expiration is sent to the SIM`
	`2-EnvelopeHandler.getTheHandler() method is called by Applet1`
		1- Applet1 is triggered
		2- No exception is thrown.	

10	**Envelope Handler availability with EVENT_CALL_CONTROL_BY_SIM**
	`1-Envelope call control by sim is sent to the SIM`
	`2-EnvelopeHandler.getTheHandler() method is called by Applet1`
		1- Applet1 is triggered
		2- No exception is thrown.	

~~10~~	~~**Envelope Handler availability with EVENT_MO_SHORT_MESSAGE_CONTROL_BY_SIM**~~ ~~`1-Envelope mo short message control by sim is sent to the SIM`~~ ~~`2-EnvelopeHandler.getTheHandler() method is called by Applet1.`~~	~~1- Applet1 is triggered~~ ~~2- No exception is throw~~	

Id	Description	API/Framework Expectation	APDU Expectation

11	**Envelope Handler availability with EVENT_MO_SHORT_MESSAGE_CONTROL_BY_SIM**
	
	`1-Envelope mo short message control by sim is sent to the SIM`
	
	`2-EnvelopeHandler.getTheHandler() method is called by Applet1.`
		1- Applet1 is triggered
		
		2- No exception is throw	

~~11~~	~~**Envelope Handler availability with EVENT_EVENT_DOWNLOAD_MT_CALL**~~ ~~`1-Envelope event download mt call is sent to the 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.~~	

12	**Envelope Handler availability with EVENT_EVENT_DOWNLOAD_MT_CALL**
	
	`1-Envelope event download mt call is sent to the 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/Framework Expectation	APDU Expectation
~~12~~	~~Envelope Handler availability with EVENT_EVENT_DOWNLOAD_CALL_CONNECTED~~		

13	**Envelope Handler availability with EVENT_EVENT_DOWNLOAD_CALL_CONNECTED**
	
	`1-Envelope event download call connected is sent to the 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.	

~~13~~	~~**Envelope Handler availability with EVENT_EVENT_DOWNLOAD_CALL_DISCONECTTED**~~ ~~`1-Envelope event download call disconnected is sent to the 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.~~	

14	**Envelope Handler availability with EVENT_EVENT_DOWNLOAD_CALL_DISCONECTTED**
	
	`1-Envelope event download call disconnected is sent to the 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.	

~~14~~	~~**Envelope Handler availability with EVENT_EVENT_DOWNLOAD_LOCATION_STATUS**~~ ~~`1-Envelope event download location status is sent to the SIM`~~	~~1- Applet1 is triggered~~	

Id	Description	API/Framework Expectation	APDU Expectation
	~~2-EnvelopeHandler.getTheHandler() method is called by Applet1~~ ~~3-EnvelopeHandler.getTheHandler() method is called by Applet2~~	~~2- No exception is thrown.~~ ~~Applet1 finalizes~~ ~~3- Applet2 is triggered~~ ~~4- No exception is thrown.~~	
15	**Envelope Handler availability with EVENT_EVENT_DOWNLOAD_LOCATION_STATUS** 1-Envelope event download location status is sent to the 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.	
~~15~~	~~**Envelope Handler availability with EVENT_EVENT_DOWNLOAD_USER_ACTIVITY**~~ ~~1-Envelope event download user activity is sent to the 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~~	
16	**Envelope Handler availability with EVENT_EVENT_DOWNLOAD_USER_ACTIVITY** 1-Envelope event download user activity is sent to the 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/Framework Expectation	APDU Expectation
~~16~~	~~Envelope Handler availability with EVENT_EVENT_DOWNLOAD_IDLE_SCREEN_AVAILABLE~~		

17	**Envelope Handler availability with EVENT_EVENT_DOWNLOAD_IDLE_SCREEN_AVAILABLE**
	
	`1-Envelope event download idle screen available is sent to the 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.	

~~17~~	~~**Envelope Handler availability with EVENT_EVENT_DOWNLOAD_CARD_READER_STATUS**~~ ~~`1-Envelope event download card reader status is sent to the 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.~~	

18	**Envelope Handler availability with EVENT_EVENT_DOWNLOAD_CARD_READER_STATUS**
	
	`1-Envelope event download card reader status is sent to the 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.	

~~18~~	~~**Envelope Handler availability with EVENT_EVENT_DOWNLOAD_LANGUAGE_**~~		

Id	Description	API/Framework Expectation	APDU Expectation
	~~SELECTION~~ ~~1-Envelope event download language selection is sent to the 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.~~	
19	**Envelope Handler availability with EVENT_EVENT_DOWNLOAD_LANGUAGE_SELECTION** 1-Envelope event download language selection is sent to the SIM 2-EnvelopeHandler.getTheHandler() method is called by Applet1 Applet1 finalizes. 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.	
~~19~~	~~**Envelope Handler availability with EVENT_EVENT_DOWNLOAD_BROWSER_TERMINATION**~~ ~~1-Envelope event download browser termination is sent to the 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.~~	
20	**Envelope Handler availability with EVENT_EVENT_DOWNLOAD_BROWSER_TERMINATION** 1-Envelope event download browser termination is sent to the 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.	
~~20~~	~~**Envelope Handler availaibility with EVENT_STATUS_COMMAND**~~ ~~1-Status command is sent to the SIM~~ ~~2-EnvelopeHandler.getTheHandler() method is called by Applet1~~	~~1- Applet1 is triggered~~ ~~2- A Toolkit exception~~	

Id	Description	API/Framework Expectation	APDU Expectation
	~~3-EnvelopeHandler.getTheHandler() method is called by Applet2~~	~~HANDLER_NOT_AVAILABLE is thrown~~ ~~Applet1 finalizes~~ ~~3- Applet2 is triggered~~ ~~4- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown~~	
~~21~~	**~~Envelope Handler availability with EVENT_ UNRECOGNIZED_ENVELOPE~~** ~~1-An unrecognized Envelope is sent to the 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/Framework Expectation	APDU Expectation
21	**Envelope Handler availability with EVENT_STATUS_COMMAND** 1-Status command is sent to the SIM 2-EnvelopeHandler.getTheHandler() method is called by Applet1 3-EnvelopeHandler.getTheHandler() method is called by Applet2	1- Applet1 is triggered 2- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown Applet1 finalizes. 3- Applet2 is triggered 4- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown	
22	**Envelope Handler availability with EVENT_EVENT_DOWNLOAD_DATA_AVAILABLE** 1- Applet1 builds a proactive command OPEN CHANNEL. proactiveHandler.send() method is called 2-Envelope event download data available is sent to the SIM with ChannelId=01. 3-EnvelopeHandler.getTheHandler() method is called by Applet1	1- Applet1 is registered to EVENT_EVENT_DOWNLOAD_DATA_AVAILABLE and EVENT_EVENT_DOWNLOAD_CHANNEL_STATUS 2- Applet1 is triggered 3-No exception is thrown.	1- OPEN CHANNEL proactive command is fetched TERMINAL RESPONSE is issued with Channel Id = 01
23	**Envelope Handler availability with EVENT_EVENT_DOWNLOAD_CHANNEL_STATUS** 1-Envelope event download channel status is sent to the SIM with ChannelId=01. 2-EnvelopeHandler.getTheHandler() method is called by Applet1	1- Applet1 is triggered 2-No exception is thrown.	
24	**Envelope Handler availability with EVENT_UNRECOGNIZED_ENVELOPE** 1-An unrecognized Envelope is sent to the 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.	
25	**Envelope Handler availability with EVENT_FORMATTED_SMS_PP_UPD** 1- A formatted Update Record EFsms instruction is sent to the SIM 2-EnvelopeHandler.getTheHandler() method is called by Applet1	1- Applet1 is triggered 2- No exception is thrown. -	
26	**Envelope Handler availability with**		

Id	Description	API/Framework Expectation	APDU Expectation
	EVENT_UNFORMATTED_SMS_PP_UPD 1-An unformatted Update Record EFsms instruction is sent to the 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.	

6.3.1.3.4 Test Coverage

CRR Number	Test Case Number
~~CRRN1~~	~~2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19~~
~~CRRC1~~	~~1, 20~~

CRRN1	3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 22, 23, 24, 25, 26
CRRC1	1, 2, 21

6.3.1.4 EnvelopeResponseHandler

Test Area Reference: ~~FWK_MHA_ ERHD.~~FWK_MHA_ERHD

6.3.1.4.1 Conformance Requirement

6.3.1.4.1.1 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_CALL_CONTROL

______EVENT_SMS_MO_CONTROL

______EVENT_UNRECOGNIZED_ENVELOPE

CRRN2: After a call to the post method the handler is not longer available

CRRN3: After a call to the send method the handler is not longer available

6.3.1.4.1.2 ~~Parameters error~~

~~No requirements.~~

~~6.3.1.4.1.3~~ Context Errors

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

EVENT_FORMATTED_SMS_CB

EVENT_UNFORMATTED_SMS_CB

EVENT_MENU_SELECTION

EVENT_MENU_SELECTION_HELP_REQUEST

EVENT_TIMER_EXPIRATION

EVENT_EVENT_DOWNLOAD_MT_CALL

EVENT_EVENT_DOWNLOAD_CALL_CONNECTED

EVENT_EVENT_DOWNLOAD_CALL_DISCONNECTED

EVENT_EVENT_DOWNLOAD_LOCATION_STATUS

EVENT_EVENT_DOWNLOAD_USER_ACTIVITY

EVENT_EVENT_DOWNLOAD_IDLE_SCREEN_AVAILABLE

EVENT_EVENT_DOWNLOAD_CARD_READER_STATUS

EVENT_EVENT_DOWNLOAD_LANGUAGE_SELECTION

EVENT_EVENT_DOWNLOAD_BROWSER_TERMINATION

EVENT_STATUS_COMMAND

EVENT_PROFILE_DOWNLOAD

EVENT_FIRST_COMMAND_AFTER_SELECT

EVENT_EVENT_DOWNLOAD_DATA_AVAILABLE

EVENT_EVENT_DOWNLOAD_CHANNEL_STATUS

EVENT_FORMATTED_SMS_PP_UPD

EVENT_UNFORMATTED_SMS_PP_UPD

6.3.1.4.2 Test Suite Files

~~Test Script: FWK_MHA_ ERHD_1.scr~~

~~Test Applet: FWK_MHA_ ERHD_1.java~~

~~FWK_MHA_ERHD_2.java~~

~~Load Script: FWK_MHA_ ERHD_1.ldr~~

~~Cleanup Script: FWK_MHA_ ERHD_1.clr~~

~~Parameter File: FWK_MHA_ ERHD_1.par~~

Test Script:	FWK_MHA_ERHD_1.scr
Test Applet:	FWK_MHA_ERHD_1.java
	FWK_MHA_ERHD_2.java
Load Script:	FWK_MHA_ERHD_1.ldr
Cleanup Script:	FWK_MHA_ERHD_1.clr

Parameter File: FWK_MHA_ERHD_1.par

6.3.1.4.3 Test Procedure

Id	Description	API/Framework Expectation	APDU Expectation
~~1~~	**~~Toolkit Applet1 and Toolkit Applet2 registration and Envelope Response Handler availability with EVENT_PROFILE_DOWNLOAD~~** ~~1- Applet1 Toolkit 1 is registered to all events defined in [7].~~ ~~Using the methods initMenuEntry() for EVENT_MENU_SELECTION, requestPollInterval() for EVENT_STATUS_COMMAND, allocateTimer() for EVENT_TIMER_EXPIRATION and setEventList() for the rest of the events.~~ ~~Applet2 Toolkit 2 is registered to EVENT_UNFORMATTED_SMS_PP_ENV and EVENT_UNRECOGNIZED_ENVELOPE.~~ ~~2 Terminal Profile command is sent to SIM without the facility of SET_EVENT_LIST ,SETUP_IDLE_MODE_TEXT, SETUP_MENU and POLL_INTERVAL.~~ ~~Applet1 is triggered~~ ~~32-EnvelopeResponseHandler.getTheHandler() method is called by Applet1~~ ~~Applet1 is deregistered to EVENT_PROFILE_DOWNLOAD~~	~~1- No exception is thrown~~ ~~2- Applet1 is triggered~~ ~~3- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown~~	
1	**Toolkit Applet1 and Toolkit Applet2 registration and Envelope Response Handler availability with EVENT_FIRST_COMMAND_AFTER_SELECT** 1- Applet1 is registered to all events defined in TS 43.019 [7]. Using the methods initMenuEntry() for EVENT_MENU_SELECTION, requestPollInterval() for EVENT_STATUS_COMMAND, allocateTimer() for EVENT_TIMER_EXPIRATION and setEventList() for the rest of the events. Applet2 is registered to EVENT_UNFORMATTED_SMS_PP_ENV and EVENT_UNRECOGNIZED_ENVELOPE. 4- Select MF. 3-EnvelopeResponseHandler.getTheHandler() method is called by Applet1 Applet1 is deregistered to EVENT_FIRST_COMMAND_AFTER_SELECT.	1- No exception is thrown 2- Applet1 is triggered by EVENT_FIRST_COMMAND_AFTER_SELECT 3- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown	
~~2~~	**~~Envelope Response Handler availability with EVENT_MENU_SELECTION_HELP_REQUEST~~** ~~Perform SIM initialization with all the facilities supported~~		

Id	Description	API/Framework Expectation	APDU Expectation
	~~1-Envelope menu selection with help request is sent to the SIM~~ ~~2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1~~	~~1- Applet1 is triggered.~~ ~~2- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown~~	
2	**Handler availability with EVENT_PROFILE_DOWNLOAD** 1- Terminal Profile command is sent to the SIM without the facility of SET_EVENT_LIST, SETUP_IDLE_MODE_TEXT, SETUP_MENU and POLL_INTERVAL. 2- EnvelopeResponseHandler.getTheHandler() method is called by Applet1 Applet1 is deregistered to EVENT_PROFILE_DOWNLOAD	1- Applet1 Is Triggered By EVENT_PROFILE_DOWNLOAD 2- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown	
3	**Envelope Response Handler availability with EVENT_MENU_SELECTION_HELP_REQUEST** Perform SIM initialization with all the facilities supported 1-Envelope menu selection with help request is sent to the SIM 2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1	1- Applet1 is triggered. 2- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown	
~~3~~	~~**Envelope Response Handler availability with EVENT_MENU_SELECTION**~~ ~~1-A envelope menu selection is sent to the SIM~~ ~~The Applet1 is triggered~~ ~~2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1~~	~~1- Applet1 is triggered~~ ~~2- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown~~	

Id	Description	API/Framework Expectation	APDU Expectation
4	**Envelope Response Handler availability with EVENT_MENU_SELECTION** 1-A envelope menu selection is sent to the SIM 2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1	1- Applet1 is triggered 2- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown	
~~4~~	~~**Envelope Response Handler availability with EVENT_UNFORMATTED_CB**~~ ~~1-Envelope cell broadcast unformatted is sent to the SIM~~ ~~2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1~~	~~1- Applet1 is triggered.~~ ~~2- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown~~	
~~5~~	~~**Envelope Response Handler availability with EVENT_TIMER_EXPIRATION**~~ ~~1-Envelope Timer Expiration is sent to the SIM~~ ~~2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1~~	~~1- Applet1 is triggered.~~ ~~2- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown~~	
5	**Envelope Response Handler availability with EVENT_FORMATTED_CB** 1-Envelope cell broadcast formatted is sent to the SIM 2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1	1- The applet1 is triggered. 2-A Toolkit exception HANDLER_NOT_AVAILABLE is thrown	
~~6~~	~~**Envelope Response Handler availability with EVENT_EVENT_DOWNLOAD_MT_CALL**~~ ~~1-Envelope event download mt call is sent to the SIM~~ ~~2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1~~	~~1- Applet1 is triggered.~~ ~~2 -A Toolkit exception HANDLER_NOT_AVAILABLE is thrown~~	
6	**Envelope Response Handler availability with EVENT_UNFORMATTED_CB** 1-Envelope cell broadcast unformatted is sent to the SIM 2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1	1- Applet1 is triggered. 2- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown	

Id	Description	API/Framework Expectation	APDU Expectation
~~7~~	~~**Envelope Response Handler availability with EVENT_EVENT_DOWNLOAD_CALL_CONNECTED**~~ ~~1-Envelope event download call connected is sent to the SIM~~ ~~2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1~~	~~1- Applet1 is triggered.~~ ~~2- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown~~	
7	**Envelope Response Handler availability with EVENT_TIMER_EXPIRATION** 1-Envelope Timer Expiration is sent to the SIM 2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1	1- Applet1 is triggered. 2- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown	
~~8~~	~~**Envelope Response Handler availability with EVENT_EVENT_DOWNLOAD_CALL_DISCONNECTED**~~ ~~1-Envelope event download call disconnected is sent to the SIM~~ ~~2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1~~	~~1- Applet1 is triggered.~~ ~~2- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown~~	
8	**Envelope Response Handler availability with EVENT_EVENT_DOWNLOAD_MT_CALL** 1-Envelope event download mt call is sent to the SIM 2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1	1- Applet1 is triggered. 2 -A Toolkit exception HANDLER_NOT_AVAILABLE is thrown	
~~9~~	~~**Envelope Response Handler availability with EVENT_EVENT_DOWNLOAD_LOCATION_STATUS**~~ ~~1-Envelope event download location status is sent to the SIM~~ ~~2-Applet A obtains the Envelope Response Handler~~	~~1- Applet1 is triggered.~~ ~~2- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown~~	
9	**Envelope Response Handler availability with EVENT_EVENT_DOWNLOAD_CALL_CONNECTED** 1-Envelope event download call connected is sent to the SIM	1- Applet1 is triggered.	

Id	Description	API/Framework Expectation	APDU Expectation
	2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1	2- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown	
~~10~~	~~**Envelope Response Handler availability with EVENT_EVENT_DOWNLOAD_USER_ACTIVITY**~~ ~~1-Envelope event download user activity is sent to the SIM~~ ~~2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1~~	~~1- Applet1 is triggered.~~ ~~2- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown~~	
10	**Envelope Response Handler availability with EVENT_EVENT_DOWNLOAD_CALL_DISCONNECTED** 1-Envelope event download call disconnected is sent to the SIM 2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1	1- Applet1 is triggered. 2- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown	
~~11~~	~~**Envelope Response Handler availability with EVENT_EVENT_DOWNLOAD_IDLE_SCREEN_AVAILABLE**~~ ~~1-Envelope event download idle screen available is sent to the SIM~~ ~~2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1~~	~~1- Applet1 is triggered.~~ ~~2- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown~~	
11	**Envelope Response Handler availability with EVENT_EVENT_DOWNLOAD_LOCATION_STATUS** 1-Envelope event download location status is sent to the SIM 2-Applet1 obtains the Envelope Response Handler	1- Applet1 is triggered. 2- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown	
~~12~~	~~**Envelope Response Handler availability with EVENT_EVENT_DOWNLOAD_CARD_READER_STATUS**~~ ~~1-Envelope event download card reader status is sent to the SIM~~ ~~2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1~~	~~1- Applet1 is triggered~~ ~~2- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown~~	

Id	Description	API/Framework Expectation	APDU Expectation
12	**Envelope Response Handler availability with EVENT_EVENT_DOWNLOAD_USER_ACTIVITY** 1-Envelope event download user activity is sent to the SIM 2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1	1- Applet1 is triggered. 2- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown	
~~13~~	~~**Envelope Response Handler availability with EVENT_STATUS_COMMAND**~~ ~~1-Status command is sent to the SIM~~ ~~2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1~~	~~1- Applet1 is triggered~~ ~~2- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown~~	

Id	Description	API/Framework Expectation	APDU Expectation

13	**Envelope Response Handler availability with EVENT_EVENT_DOWNLOAD_IDLE_SCREEN_AVAILABLE**
	`1-Envelope event download idle screen available is sent to the SIM`
	`2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1`
		1- Applet1 is triggered.
		2- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown	

~~14~~	~~**Envelope Response Handler availability with EVENT_FORMATTED_SMS_PP_ENV**~~ ~~`1-A formatted sms pp envelope is sent to the SIM`~~ ~~`2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1`~~ ~~`3-Applet1 builds an additional information for response packet and it calls the post method`~~ ~~`4-Applet1 calls all methods of the Envelope Response Handler (including the inherited method)`~~ ~~`The Applet1 finalizes`~~ ~~`5-A EVENT_FORMATTED_SMS_PP_ENV envelope is sent to the 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 Envelope Response Handler (including the inherited method)`~~	~~1- Applet1 is triggered~~ ~~2- No exception is thrown.~~ ~~4- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown for each method~~ ~~5- Applet1 is triggered~~ ~~6- No Exception is thrown~~ ~~8- Toolkit exception HANDLER_NOT_AVAILABLE is thrown for each method~~	~~3- The response packet is sent~~ ~~7- The proactive command is sent~~

14	**Envelope Response Handler availability with EVENT_EVENT_DOWNLOAD_CARD_READER_STATUS**
	`1-Envelope event download card reader status is sent to the SIM`
	`2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1`
		1- Applet1 is triggered
		2- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown	

Id	Description	API/Framework Expectation	APDU Expectation
~~15~~	~~**Envelope Response Handler availability with EVENT_UNFORMATTED_SMS_PP_ENV**~~		

15	**Envelope Response Handler availability with EVENT_EVENT_DOWNLOAD_LANGUAGE_SELECTION**
	
	`1-Envelope event download language selection is sent to the SIM`
	
	`2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1`
		1- Applet1 is triggered
		
		2-A Toolkit exception HANDLER_NOT_AVAILABLE is thrown	

Id	Description	API/Framework Expectation	APDU Expectation
~~16~~	~~**Envelope Response Handler availability with EVENT_CALL_CONTROL_BY_SIM**~~		
16	**Envelope Response Handler availability with EVENT_EVENT_DOWNLOAD_BROWSER_TERMINATION** 1-Envelope event download browser termination is sent to the SIM 2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1	1- Applet1 is triggered 2-A Toolkit exception HANDLER_NOT_AVAILABLE is thrown	
~~17~~	~~**Envelope Response Handler availability with EVENT_MO_SHORT_MESSAGE_CONTROL_BY_SIM**~~ ~~1-Envelope mo short message control by sim is sent to the 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 Envelope Response Handler (including the inherited method)~~ ~~The Applet1 finalizes~~ ~~5-Envelope mo short message control by sim is sent to the SIM~~ ~~6-EnvelopeResponseHandler.getTheHandler() method is called by Applet1~~ ~~7-Applet1 builds a proactive command and it calls the send method~~	~~1- Applet1 is triggered~~ ~~2- No exception is thrown.~~ ~~4- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown for each method~~ ~~5- Applet1 is triggered~~ ~~6- No exception is thrown~~	~~3-The envelope response is sent~~ ~~7- The proactive command~~

Id	Description	API/Framework Expectation	APDU Expectation
	~~8-Applet1 calls all methods of the Envelope Response Handler (including the inherited method)~~	~~8- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown for each method~~	~~is fetched and the Terminal Response is issued~~
17	**Envelope Response Handler availability with EVENT_STATUS_COMMAND** 1-Status command is sent to the SIM 2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1	1- Applet1 is triggered 2- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown	

Id	Description	API/Framework Expectation	APDU Expectation
~~18~~	~~**Envelope Response Handler availability with EVENT_UNRECOGNIZED_ENVELOPE**~~		

Id	Description	API/Framework Expectation	APDU Expectation
18	**Envelope Response Handler availability with EVENT_EVENT_DOWNLOAD_DATA_AVAILABLE** 1- Applet1 initialises a proactive command OPEN CHANNEL and calls the send() method. 2- Envelope event download data avalaible is sent to the SIM with channelId=01 3-EnvelopeResponseHandler.getTheHandler() method is called by Applet1	2- Applet1 is triggered 3- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown	1- The OPEN CHANNEL command is fetched. TERMINAL RESPONSE IS SENT TO THE SIM with channelId=01
~~19~~	~~**The envelope response is sent when a proactive session is ongoing**~~ ~~1-A formatted SMS PP envelope is sent to the SIM.~~ ~~2-Proactive command DISPLAY TEXT is built and it calls the send() method.~~ ~~3-A call control by sim envelope is sent to the SIM.~~ ~~4-EnvelopeResponseHandler.getTheHandler() method is called by Applet1~~ ~~5-Applet1 builds the envelope response and it calls the postAsBERTLV~~	~~1- Applet1 is triggered.~~ ~~3- Applet1 is triggered~~ ~~4- No exception is thrown~~	~~2- 91 XX~~ ~~5-The envelope response is sent~~ ~~9F YY~~ ~~GET RESPONSE~~ ~~Data~~ ~~91 XX~~ ~~Fetch DISPLAY TEXT~~ ~~Terminal Response DISPLAY TEXT~~
~~NOTE: Due to an inconsistency in GSM 03.19 [7] specification it is not possible to cover the test case when an applet try to post data in multitriggering.~~			
19	**Envelope Response Handler availability with EVENT_EVENT_DOWNLOAD_CHANNEL_STATUS** 1- Envelope event download channel status is sent to the SIM with channelId=01 2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1	1- Applet1 is triggered 2- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown	
20	**Envelope Response Handler availability with EVENT_FORMATTED_SMS_PP_UPD** 1- Update Record EFsms instruction formatted is sent to the SIM	1- The applet1 is triggered.	

Id	Description	API/Framework Expectation	APDU Expectation
	2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1	2-A Toolkit exception HANDLER_NOT_AVAILABLE is thrown	

Id	Description	API/Framework Expectation	APDU Expectation
21	**Envelope Response Handler availability with EVENT_UNFORMATTED_SMS_PP_UPD** 1- Update Record EFsms instruction unformatted is sent to the SIM 2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1	1- Applet1 is triggered. 2- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown	
22	**Envelope Response Handler availability with EVENT_FORMATTED_SMS_PP_ENV** 1-A formatted sms pp envelope is sent to the SIM 2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1 3-Applet1 builds an additional information for response packet and it calls the post method 4-Applet1 calls all methods of the Envelope Response Handler (including the inherited method) 5-A EVENT_FORMATTED_SMS_PP_ENV envelope is sent to the 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 Envelope Response Handler (including the inherited method)	1- Applet1 is triggered 2- No exception is thrown. 4- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown for each method Applet1 finalizes 5- Applet1 is triggered 6- No Exception is thrown 8- Toolkit exception HANDLER_NOT_AVAILABLE is thrown for each method	3- The response packet is sent 7- The proactive command is sent
23	**Envelope Response Handler availability with EVENT_UNFORMATTED_SMS_PP_ENV** 1-An unformatted sms pp envelope is sent to the SIM 2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1 3-Applet1 builds the envelope response and it calls the post() method 4- Applet1 calls all methods of the Envelope Response Handler (including the inherited method) 5-EnvelopeResponseHandler.getTheHandler() method is called	1- Applet1 is triggered 2- No exception is thrown. 4- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown for each method Applet1 finalizes 5- Applet2 is triggered. A Toolkit exception HANDLER_NOT_AVAILABLE is thrown.	3- The envelope response is sent

Id	Description	API/Framework Expectation	APDU Expectation
	6-An unformatted sms pp envelope is sent to the SIM 7-EnvelopeResponseHandler.getTheHandler() method is called. 8-Applet1 builds a proactive command and it calls the send() method 9-Applet1 calls all methods of the Envelope Response Handler (including the inherited method) 10-EnvelopeResponseHandler.getTheHandler() method is called by Applet2	Applet2 finalizes 6- Applet1 is triggered. 7- No exception is thrown. 9- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown for each method. Applet1 finalizes 10- Applet2 is triggered. A Toolkit exception HANDLER_NOT_AVAILABLE is thrown.	9- The proactive command is fetched and the Terminal response is issued.

Id	Description	API/Framework Expectation	APDU Expectation
24	**Envelope Response Handler availability with EVENT_CALL_CONTROL_BY_SIM** 1-Envelope call control by sim is sent to the 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 Envelope Response Handler (including the inherited method) 5-Envelope call control by sim is sent to the 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 Envelope Response Handler (including the inherited method)	1- Applet1 is triggered 2- No exception is thrown. 4- Toolkit exception HANDLER_NOT_AVAILABLE is thrown for each method Applet1 finalizes 5- Applet1 is triggered 6- No Exception is thrown 8- A Toolkit exception 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
25	**Envelope Response Handler availability with EVENT_MO_SHORT_MESSAGE_CONTROL_BY_SIM** 1-Envelope mo short message control by sim is sent to the 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 Envelope Response Handler (including the inherited method) 5-Envelope mo short message control by sim is sent to the 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 Envelope Response Handler (including the inherited method)	1- Applet1 is triggered 2- No exception is thrown. 4- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown for each method Applet1 finalizes 5- Applet1 is triggered 6- No exception is thrown 8- A Toolkit exception 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
26	**Envelope Response Handler availability with EVENT_UNRECOGNIZED_ENVELOPE** 1-An unrecognized Envelope is sent to the SIM 2-EnvelopeResponseHandler.getTheHandler()	1- Applet1 is triggered	

Id	Description	API/Framework Expectation	APDU Expectation
	method is called by Applet1 3-Applet1 builds the envelope response and it calls the postAsBERTLV() or post method 4-Applet1 calls all methods of Envelope Response Handler (including the inherited method) 5-EnvelopeResponseHandler.getTheHandler() method is called 6-An unrecognized Envelope is sent to the SIM 7-EnvelopeResponseHandler.getTheHandler() method is called 8-Applet1 builds a proactive command and it calls the send() method 9-Applet1 calls all methods of the Envelope Response Handler (including the inherited method) 10-EnvelopeResponseHandler.getTheHandler() method is called by Applet2	2- No exception is thrown. 4- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown for each method Applet1 finalizes 5- Applet2 is triggered. A Toolkit exception HANDLER_NOT_AVAILABLE is thrown. Applet2 finalizes 6- Applet1 is triggered. 7- No exception is thrown. 9- A Toolkit exception HANDLER_NOT_AVAILABLE is thrown for each method Applet1 finalizes 10- Applet2 is triggered A Toolkit exception HANDLER_NOT_AVAILABLE is thrown.	3- The envelope response is sent 9- The proactive command is fetched and the Terminal response is issued

Id	Description	API/Framework Expectation	APDU Expectation

27	**The envelope response is sent when a proactive session is ongoing**
	
	`1-A formatted SMS PP envelope is sent to the SIM.`
	
	`2-Proactive command DISPLAY TEXT is built and it calls the send() method.`
	
	`3-A call control by sim envelope is sent to the SIM.`
	
	`4-EnvelopeResponseHandler.getTheHandler() method is called by Applet1`
	
	`5-Applet1 builds the envelope response and it calls the postAsBERTLV`
		1- Applet1 is triggered.
		
		3- Applet1 is triggered
		
		4- No exception is thrown
			2- 91 XX
			
			5-The envelope response is sent
			9F YY
			
			GET RESPONSE
			Data
			91 XX
			Fetch DISPLAY TEXT
			
			Terminal Response
			DISPLAY TEXT

28	**Envelope Response Handler availability with EVENT_UNFORMATTED_SMS_PP_ENV in case of multi-triggering**
	
	`1-A unformatted sms pp envelope is sent to the 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.

29	**Envelope Response Handler availability with EVENT_UNRECOGNIZED_ENVELOPE in case of multi-triggering**
	
	`1-An unrecognized Envelope is sent to the SIM`
	
	`2-EnvelopeResponseHandler.getTheHandler() method is called by Applet1`
		1- Applet1 is triggered
		
		2- No exception is thrown.
		
		3- Applet1 finalizes	

Id	Description	API/Framework Expectation	APDU Expectation
	5-EnvelopeResponseHandler.getTheHandler() method is called by Applet 2 6- Applet2 calls the post( ) method	4- Applet2 is triggered. 5- No Exception is thrown Applet2 finalizes	6- The response is checked

~~6.3.1.4.4 Test Coverage~~

~~CRR Number~~	~~Test Case Number~~
~~CRRN1~~	~~14, 15, 16, 17, 18, 19~~
~~CRRN2~~	~~14, 15, 16, 17, 18, 19~~
~~CRRN3~~	~~14, 15, 16, 17, 18, 19~~
~~CRRC1~~	~~1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13~~

6.3.1.4.4 Test Coverage

CRR Number	Test Case Number
CRRN1	20, 21, 22, 23, 24, 25,26,27
CRRN2	20, 21, 22, 23, 24, 25
CRRN3	20, 21, 22, 23, 24, 25
CRRC1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 28, 29

6.3.2 Handler Integrity

6.3.2.3 EnvelopeHandler

Test Area Reference: FWK_HIN_ENHD

6.3.2.3.1 Conformance Requirement

6.3.2.3.1.1 Normal Execution

CRRN1: The EnvelopeHandler and its content are available for all triggered toolkit applets, from the invocation to the termination of their processToolkit method.

CRRN2: The SIM Toolkit Framework guarantees that all triggered toolkit applets receive the data.

CRRN3: The SIM Toolkit Framework shall convert the Update Record EFsms in the EnvelopeHandler TLV List containing Device Identities TLV, Address TLV and SMS TPDU TLV.

CRRN4: The getEnvelopeTag() method shall return *BTAG_SMS_PP_DOWNLOAD.*

CRRN5: The getLength() method shall return the Simple TLV list length.

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

6.3.2.3.2 Test Suite Files

Test Script:	FWK_HIN_ENHD_1.scr
Test Applet:	FWK_HIN_ENHD_1.java
Load Script:	FWK_HIN_ENHD_1.ldr
Cleanup Script:	FWK_HIN_ENHD_1.clr
Parameter File:	FWK_HIN_ENHD_1.par

6.3.2.3.3 Test Procedure

Id	Description	API/Framework Expectation	APDU Expectation
~~1~~	**~~Applet initialization and Envelope Handler integrity checks with EVENT_MENU_SELECTION_HELP_REQUEST~~** ~~1- Applet is registered to all events defined in [7] except EVENT_PROFILE_DOWNLOAD and EVENT_STATUS_COMMAND. Using the methods initMenuEntry() for EVENT_MENU_SELECTION, allocateTimer()for EVENT_TIMER_EXPIRATION, and setEventList() for the rest of the events. Perform SIM initialization with all the facilities supported~~ ~~2-Envelope menu selection with help request is sent to the SIM~~ ~~3-EnvelopeHandler.getTheHandler() method is called~~ ~~4-Copy the contents of the envelope handler in buffer 1 using EnvelopeHandler.copy()~~ ~~The EnvelopeHandler.findTLV() method is called with TAG_HELP_REQUEST~~ ~~5-A proactive command DISPLAY TEXT is sent~~ ~~6-Envelope call control by sim is sent to SIM~~ ~~EnvelopeHandler.getTheHandler() method is called~~ ~~7- It's checked that the contents of the envelope handler is 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_HELP_REQUEST is the TLV selected~~	~~1-No exception is thrown~~ ~~2- Applet is triggered~~ ~~3-No exception is thrown.~~ ~~4-No exception is thrown~~ ~~6- Applet is triggered~~ ~~7-No exception is thrown and the handler contains the envelope call control by SIM~~	~~5-91 xx.~~ ~~A proactive command Display Text is fetched~~ ~~The terminal Response of DISPLAY TEXT is sent to the SIM~~

Id	Description	API/Framework Expectation	APDU Expectation
	~~8-The contents of EnvelopeHandler are compared with buffer1 using Util.arrayCompare()~~	~~8-The contents of the envelope handler shall be the same as stored in buffer 1~~	
1	**Applet initialization and Envelope Handler integrity checks with EVENT_MENU_SELECTION_HELP_REQUEST** 1- Applet is registered to all events defined in TS 43.019 [7] except EVENT_PROFILE_DOWNLOAD and EVENT_STATUS_COMMAND. Using the methods initMenuEntry() for EVENT_MENU_SELECTION, allocateTimer()for EVENT_TIMER_EXPIRATION, and setEventList() for the rest of the events. Perform SIM initialization with all the facilities supported 2-Envelope menu selection with help request is sent to the SIM 3-EnvelopeHandler.getTheHandler() method is called 4-Copy the contents of the envelope handler in buffer 1 using EnvelopeHandler.copy() The EnvelopeHandler.findTLV() method is called with TAG_HELP_REQUEST 5-A proactive command DISPLAY TEXT is sent 6-Envelope call control by sim is sent to SIM EnvelopeHandler.getTheHandler() method is called 7- It's checked that the contents of the envelope handler is 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_HELP_REQUEST is the TLV selected 8-The contents of EnvelopeHandler are compared with buffer1 using Util.arrayCompare()	1-No exception is thrown 2- Applet is triggered 3- No exception is thrown. 4- No exception is thrown 6- Applet is triggered 7- No exception is thrown and the handler contains the envelope call control by SIM 8- The contents of the envelope handler shall be the same as stored in buffer 1	5- 91 xx. A proactive command Display Text is fetched The terminal Response of DISPLAY TEXT is sent to the SIM
~~2~~	~~**Envelope Handler integrity checks with EVENT_MENU_SELECTION**~~ ~~1-An envelope menu selection is sent to SIM~~	~~1-Applet is triggered~~	

Id	Description	API/Framework Expectation	APDU Expectation
	~~2-EnvelopeHandler.getTheHandler() method is called~~ ~~3-Copy the contents of the envelope handler in buffer 1 using EnvelopeHandler.copy()~~ ~~The EnvelopeHandler.findTLV() method is called with TAG_ITEM_IDENTIFIER~~ ~~4-A proactive command DISPLAY TEXT is sent~~ ~~5-Envelope call control by sim is sent to SIM~~ ~~EnvelopeHandler.getTheHandler() method is called~~ ~~6- It's checked the contents of the envelope handler is 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.~~ ~~It's checked that the TAG_ITEM_IDENTIFIER is the TLV selected~~ ~~7- The contents of EnvelopeHandler are compared with buffer1 using Util.arrayCompare()~~	~~2-No exception is thrown.~~ ~~3-No exception is thrown.~~ ~~5Applet is triggered~~ ~~6- No exception is thrown and the handler contains the envelope call control by SIM~~ ~~7- The contents of the envelope handler shall be the same as stored in buffer 1~~	~~4-91 XX~~ ~~Proactive command Display Text is fetched~~ ~~The terminal Response of DISPLAY TEXT is sent to the SIM~~
2	**Envelope Handler integrity checks with EVENT_MENU_SELECTION** 1-An envelope menu selection is sent to SIM 2-EnvelopeHandler.getTheHandler() method is called 3-Copy the contents of the envelope handler in buffer 1 using EnvelopeHandler.copy() The EnvelopeHandler.findTLV() method is called with TAG_ITEM_IDENTIFIER 4-A proactive command DISPLAY TEXT is sent 5-Envelope call control by sim is sent to SIM EnvelopeHandler.getTheHandler() method is called 6- It's checked the contents of the envelope handler is the envelope call control using EnvelopeHandler.copy() and Util.arrayCompare() methods	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	4- 91 XX

Id	Description	API/Framework Expectation	APDU Expectation
	The EnvelopeHandler.findTLV() method is called with TAG_DEVICE_IDENTITIES Call Control execution is finished. It's checked that the TAG_ITEM_IDENTIFIER is the TLV selected 7- The contents of EnvelopeHandler are compared with buffer1 using Util.arrayCompare()	control by SIM 7- The contents of the envelope handler shall be the same as stored in buffer 1	Proactive command Display Text is fetched The terminal Response of DISPLAY TEXT is sent to the SIM

Id	Description	API/Framework Expectation	APDU Expectation
~~3~~	~~**Envelope Handler integrity checks with EVENT_FORMATTED_SMS_PP_ENV**~~ ~~1-A formatted sms pp envelope is sent to SIM~~ ~~2-EnvelopeHandler.getTheHandler() method is called~~ ~~3-Copy the contents of the envelope handler in buffer 1 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 SIM~~ ~~EnvelopeHandler.getTheHandler() method is called~~ ~~6-It's checked that the contents of the envelope handler is 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.~~ ~~It's checked that the TAG_SMS_TPDU is the TLV selected~~ ~~7- The contents of EnvelopeHandler are 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 envelope handler shall be the same as stored in buffer 1~~	~~4-91 XX~~ ~~Proactive command Display Text is fetched~~ ~~The terminal Response of DISPLAY TEXT is sent to the SIM~~
3	**Envelope Handler integrity checks with EVENT_FORMATTED_SMS_PP_ENV** 1-A formatted sms pp envelope is sent to SIM 2-EnvelopeHandler.getTheHandler() method is called 3-Copy the contents of the envelope handler in buffer 1 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 SIM EnvelopeHandler.getTheHandler() method is called 6-It's checked that the contents of the	1- Applet is triggered 2- No exception is thrown. 3- No exception is thrown. 5- Applet is triggered	4- 91 XX

Id	Description	API/Framework Expectation	APDU Expectation
	envelope handler is 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. It's checked that the TAG_SMS_TPDU is the TLV selected 7- The contents of EnvelopeHandler are compared with buffer1 using Util.arrayCompare()	6- No exception is thrown and the handler contains the envelope call control by SIM 7- The contents of the envelope handler shall be the same as stored in buffer 1	Proactive command Display Text is fetched The terminal Response of DISPLAY TEXT is sent to the SIM

Id	Description	API/Framework Expectation	APDU Expectation
~~4~~	~~**Envelope Handler integrity checks with EVENT_UNFORMATTED_SMS_PP_ENV**~~ ~~1-A unformatted sms pp envelope is sent to SIM~~ ~~2-EnvelopeHandler.getTheHandler() method is called~~ ~~3-Copy the contents of the envelope handler in buffer 1 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 SIM~~ ~~EnvelopeHandler.getTheHandler() method is called~~ ~~6-It's checked that the contents of the envelope handler is 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.~~ ~~It's checked that the TAG_DEVICE_IDENTITIES is the TLV selected~~ ~~7- The contents of EnvelopeHandler are 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 envelope handler shall be the same as stored in buffer 1.~~	~~4-91 XX~~ ~~Proactive command Display Text is fetched~~ ~~The terminal Response of DISPLAY TEXT is sent to the SIM~~
4	**Envelope Handler integrity checks with EVENT_UNFORMATTED_SMS_PP_ENV** 1-A unformatted sms pp envelope is sent to SIM 2-EnvelopeHandler.getTheHandler() method is called 3-Copy the contents of the envelope handler in buffer 1 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 SIM EnvelopeHandler.getTheHandler() method is called	1- Applet is triggered 2- No exception is thrown. 3- No exception is thrown. 5- Applet is triggered	4- 91 XX

Id	Description	API/Framework Expectation	APDU Expectation
	6-It's checked that the contents of the envelope handler is 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. It's checked that the TAG_DEVICE_IDENTITIES is the TLV selected 7- The contents of EnvelopeHandler are compared with buffer1 using Util.arrayCompare()	6- No exception is thrown and the handler contains the envelope call control by SIM 7- The contents of the envelope handler shall be the same as stored in buffer 1.	Proactive command Display Text is fetched The terminal Response of DISPLAY TEXT is sent to the SIM
~~5~~	~~**Envelope Handler integrity checks with EVENT_UNFORMATTED_SMS_CB**~~ ~~1-A unformatted cellbroadcast envelope is sent to SIM~~ ~~2-EnvelopeHandler.getTheHandler() method is called~~ ~~3-Copy the contents of the envelope handler in buffer 1 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 SIM~~ ~~EnvelopeHandler.getTheHandler() method is called~~ ~~6-It's checked that the contents of the envelope handler is 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.~~ ~~It's checked that the TAG_CELLBROADCAST_PAGE is the TLV selected~~ ~~7- The contents of EnvelopeHandler are 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 envelope handler shall be the same as stored in buffer 1.~~	~~4-91 XX~~ ~~Proactive command Display Text is fetched~~ ~~The terminal Response of DISPLAY TEXT is sent to the SIM~~

Id	Description	API/Framework Expectation	APDU Expectation
5	**Envelope Handler integrity checks with EVENT_UNFORMATTED_SMS_CB** 1-A unformatted cellbroadcast envelope is sent to SIM 2-EnvelopeHandler.getTheHandler() method is called 3-Copy the contents of the envelope handler in buffer 1 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 SIM EnvelopeHandler.getTheHandler() method is called 6-It's checked that the contents of the envelope handler is 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. It's checked that the TAG_CELLBROADCAST_PAGE is the TLV selected 7- The contents of EnvelopeHandler are 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 envelope handler shall be the same as stored in buffer 1.	4- 91 XX Proactive command Display Text is fetched The terminal Response of DISPLAY TEXT is sent to the SIM

Id	Description	API/Framework Expectation	APDU Expectation
~~6~~	~~**Envelope Handler integrity checks with EVENT_TIMER_EXPIRATION**~~ ~~1-A timer expiration envelope is sent to SIM~~ ~~2-EnvelopeHandler.getTheHandler() method is called~~ ~~3-Copy the contents of the envelope handler in buffer 1 using EnvelopeHandler.copy()~~ ~~The EnvelopeHandler.findTLV() method is called with TAG_TIMER_ID~~ ~~4-A proactive command DISPLAY TEXT is sent~~ ~~5-Envelope call control by sim is sent to SIM~~ ~~EnvelopeHandler.getTheHandler() method is called~~ ~~6-It's checked that the contents of the envelope handler is 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.~~ ~~It's checked that the TAG_TIMER_ID is the TLV selected~~ ~~7- The contents of EnvelopeHandler are 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 envelope handler shall be the same as stored in buffer 1~~	~~4-91 XX~~ ~~Proactive command Display Text is fetched~~ ~~The terminal Response of DISPLAY TEXT is sent to the SIM~~
6	**Envelope Handler integrity checks with EVENT_TIMER_EXPIRATION** 1-A timer expiration envelope is sent to SIM 2-EnvelopeHandler.getTheHandler() method is called 3-Copy the contents of the envelope handler in buffer 1 using EnvelopeHandler.copy() The EnvelopeHandler.findTLV() method is called with TAG_TIMER_ID 4-A proactive command DISPLAY TEXT is sent 5-Envelope call control by sim is sent to SIM EnvelopeHandler.getTheHandler() method is	1- Applet is triggered 2- No exception is thrown. 3- No exception is thrown. 5- Applet is triggered	4- 91 XX

Id	Description	API/Framework Expectation	APDU Expectation
	called 6-It’s checked that the contents of the envelope handler is 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. It’s checked that the TAG_TIMER_ID is the TLV selected 7- The contents of EnvelopeHandler are compared with buffer1 using Util.arrayCompare()	6- No exception is thrown and the handler contains the envelope call control by SIM 7- The contents of the envelope handler shall be the same as stored in buffer 1	Proactive command Display Text is fetched The terminal Response of DISPLAY TEXT is sent to the SIM
~~7~~	~~**Envelope Handler integrity checks with EVENT_CALL_CONTROL_BY_SIM**~~ ~~1-A call control envelope is sent to SIM~~ ~~2-EnvelopeHandler.getTheHandler() method is called~~ ~~3-Copy the contents of the envelope handler in buffer 1 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 SIM~~ ~~EnvelopeHandler.getTheHandler() method is called~~ ~~6-It’s checked that the contents of the envelope handler is 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.~~ ~~It’s checked that the TAG_ADDRESS is the TLV selected~~ ~~7- The contents of EnvelopeHandler are 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 envelope handler shall be the same as stored in buffer 1~~	~~4-91 XX~~ ~~Proactive command Display Text is fetched~~ ~~The terminal Response of DISPLAY TEXT is sent to the SIM~~

Id	Description	API/Framework Expectation	APDU Expectation
7	**Envelope Handler integrity checks with EVENT_CALL_CONTROL_BY_SIM** 1-A call control envelope is sent to SIM 2-EnvelopeHandler.getTheHandler() method is called 3-Copy the contents of the envelope handler in buffer 1 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 SIM EnvelopeHandler.getTheHandler() method is called 6-It's checked that the contents of the envelope handler is 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. It's checked that the TAG_ADDRESS is the TLV selected 7- The contents of EnvelopeHandler are 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 envelope handler shall be the same as stored in buffer 1	4- 91 XX Proactive command Display Text is fetched The terminal Response of DISPLAY TEXT is sent to the SIM
~~8~~	~~**Envelope Handler integrity checks with EVENT_MO_SHORT_MESSAGE_CONTROL_BY_SIM**~~ ~~1-A mo short message control by sim envelope is sent to SIM~~ ~~2-EnvelopeHandler.getTheHandler() method is called~~ ~~3-Copy the contents of the envelope handler in buffer 1 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 SIM~~ ~~EnvelopeHandler.getTheHandler() method is called~~	~~1- Applet is triggered~~ ~~2-No exception is thrown.~~ ~~3-No exception is thrown.~~ ~~5- Applet is triggered~~	~~4-91 XX~~

Id	Description	API/Framework Expectation	APDU Expectation
	~~6-It's checked that the contents of the envelope handler is 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.~~ ~~It's checked that the TAG_ADDRESS is the TLV selected~~ ~~7- The contents of EnvelopeHandler are compared with buffer1 using Util.arrayCompare()~~	~~6- No exception is thrown and the handler contains the envelope call control by SIM~~ ~~7- The contents of the envelope handler shall be the same as stored in buffer 1.~~	~~Proactive command Display Text is fetched~~ ~~The terminal Response of DISPLAY TEXT is sent to the SIM~~
8	**Envelope Handler integrity checks with EVENT_ MO_SHORT_MESSAGE_CONTROL_BY_SIM** 1-A mo short message control by sim envelope is sent to SIM 2-EnvelopeHandler.getTheHandler() method is called 3-Copy the contents of the envelope handler in buffer 1 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 SIM EnvelopeHandler.getTheHandler() method is called 6-It's checked that the contents of the envelope handler is 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. It's checked that the TAG_ADDRESS is the TLV selected 7- The contents of EnvelopeHandler are 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 envelope handler shall be the same as stored in buffer 1.	4- 91 XX Proactive command Display Text is fetched The terminal Response of DISPLAY TEXT is sent to the SIM

Id	Description	API/Framework Expectation	APDU Expectation
~~9~~	**~~Envelope Handler integrity checks with EVENT_ EVENT_DOWNLOAD_MT_CALL~~** ~~1-A event download mt call envelope is sent to SIM~~ ~~2-EnvelopeHandler.getTheHandler() method is called~~ ~~3-Copy the contents of the envelope handler in buffer 1 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 SIM~~ ~~EnvelopeHandler.getTheHandler() method is called~~ ~~6-It's checked that the contents of the envelope handler is 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.~~ ~~It's checked that the TAG_ADDRESS is the TLV selected~~ ~~7- The contents of EnvelopeHandler are compared with buffer1 using Util.arrayCompare()~~	~~1-~~ ~~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 envelope handler shall be the same as stored in buffer 1~~	~~4-91 XX~~ ~~Proactive command Display Text is fetched~~ ~~The terminal Response of DISPLAY TEXT is sent to the SIM~~
9	**Envelope Handler integrity checks with EVENT_ EVENT_DOWNLOAD_MT_CALL** 1-A event download mt call envelope is sent to SIM 2-EnvelopeHandler.getTheHandler() method is called 3-Copy the contents of the envelope handler in buffer 1 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 SIM	1- Applet is triggered 2- No exception is thrown. 3- No exception is thrown. 5- Applet is triggered	4- 91 XX

Id	Description	API/Framework Expectation	APDU Expectation
	EnvelopeHandler.getTheHandler() method is called 6-It's checked that the contents of the envelope handler is 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. It's checked that the TAG_ADDRESS is the TLV selected 7- The contents of EnvelopeHandler are compared with buffer1 using Util.arrayCompare()	6- No exception is thrown and the handler contains the envelope call control by SIM 7- The contents of the envelope handler shall be the same as stored in buffer 1	Proactive command Display Text is fetched The terminal Response of DISPLAY TEXT is sent to the SIM
~~10~~	~~**Envelope Handler integrity checks with EVENT_ EVENT_DOWNLOAD_CALL_CONNECTED**~~ ~~1-A event download call connected envelope is sent to SIM~~ ~~2-EnvelopeHandler.getTheHandler() method is called~~ ~~3-Copy the contents of the envelope handler in buffer 1 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 SIM~~ ~~EnvelopeHandler.getTheHandler() method is called~~ ~~6-It's checked that the contents of the envelope handler is 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.~~ ~~It's checked that the TAG_ADDRESS is the TLV selected~~	~~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~~	~~4-91 XX~~ ~~Proactive command Display Text is fetched~~ ~~The terminal Response of DISPLAY TEXT is sent to the SIM~~

Id	Description	API/Framework Expectation	APDU Expectation
	~~7- The contents of EnvelopeHandler are compared with buffer1 using Util.arrayCompare()~~	~~7- The contents of the envelope handler shall be the same as stored in buffer 1.~~	
10	**Envelope Handler integrity checks with EVENT EVENT_DOWNLOAD_CALL_CONNECTED** 1-A event download call connected envelope is sent to SIM 2-EnvelopeHandler.getTheHandler() method is called 3-Copy the contents of the envelope handler in buffer 1 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 SIM EnvelopeHandler.getTheHandler() method is called 6-It's checked that the contents of the envelope handler is 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. It's checked that the TAG_ADDRESS is the TLV selected 7- The contents of EnvelopeHandler are 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 envelope handler shall be the same as stored in buffer 1.	4- 91 XX Proactive command Display Text is fetched The terminal Response of DISPLAY TEXT is sent to the SIM
~~11~~	~~**Envelope Handler integrity checks with EVENT EVENT_DOWNLOAD_CALL_DISCONNECTED**~~ ~~1-A event download call disconnected envelope is sent to SIM~~ ~~2-EnvelopeHandler.getTheHandler() method is called~~ ~~3-Copy the contents of the envelope handler in buffer 1 using EnvelopeHandler.copy()~~ ~~The EnvelopeHandler.findTLV() method is called with TAG_ADDRESS~~ ~~4-A proactive command DISPLAY TEXT is sent~~	~~1- Applet is triggered~~ ~~2-No exception is thrown.~~ ~~3-No exception is thrown.~~	~~4-91 XX~~

Id	Description	API/Framework Expectation	APDU Expectation
	~~5-Envelope call control by sim is sent to SIM~~ ~~EnvelopeHandler.getTheHandler() method is called~~ ~~6-It's checked that the contents of the envelope handler is 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.~~ ~~It's checked that the TAG_ADDRESS is the TLV selected~~ ~~7- The contents of EnvelopeHandler are compared with buffer1 using Util.arrayCompare()~~	~~5- Applet is triggered~~ ~~6- No exception is thrown and the handler contains the envelope call control by SIM~~ ~~7- The contents of the envelope handler shall be the same as stored in buffer 1.~~	~~Proactive command Display Text is fetched~~ ~~The terminal Response of DISPLAY TEXT is sent to the SIM~~
11	**Envelope Handler integrity checks with EVENT EVENT_DOWNLOAD_CALL_DISCONNECTED** 1-A event download call disconnected envelope is sent to SIM 2-EnvelopeHandler.getTheHandler() method is called 3-Copy the contents of the envelope handler in buffer 1 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 SIM EnvelopeHandler.getTheHandler() method is called 6-It's checked that the contents of the envelope handler is 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. It's checked that the TAG_ADDRESS is the TLV selected	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	4- 91 XX Proactive command Display Text is fetched The terminal Response of DISPLAY TEXT is sent to the SIM

Id	Description	API/Framework Expectation	APDU Expectation
	7- The contents of EnvelopeHandler are compared with buffer1 using Util.arrayCompare()	7- The contents of the envelope handler shall be the same as stored in buffer 1.	
~~12~~	~~**Envelope Handler integrity checks with EVENT_ EVENT_DOWNLOAD_LOCATION_STATUS**~~ ~~1-A event download location status envelope is sent to SIM~~ ~~2-EnvelopeHandler.getTheHandler() method is called~~ ~~3-Copy the contents of the envelope handler in buffer 1 using EnvelopeHandler.copy()~~ ~~The EnvelopeHandler.findTLV() method is called with TAG_LOCATION_STATUS~~ ~~4-A proactive command DISPLAY TEXT is sent~~ ~~5-Envelope call control by sim is sent to SIM~~ ~~EnvelopeHandler.getTheHandler() method is called~~ ~~6-It's checked that the contents of the envelope handler is 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.~~ ~~It's checked that the TAG_LOCATION_STATUS is the TLV selected~~ ~~7- The contents of EnvelopeHandler are 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 envelope handler shall be the same as stored in buffer 1~~	~~4-91 XX~~ ~~Proactive command Display Text is fetched~~ ~~The terminal Response of DISPLAY TEXT is sent to the SIM~~
12	**Envelope Handler integrity checks with EVENT_ EVENT_DOWNLOAD_LOCATION_STATUS** 1-A event download location status envelope is sent to SIM 2-EnvelopeHandler.getTheHandler() method is called 3-Copy the contents of the envelope handler in buffer 1 using EnvelopeHandler.copy() The EnvelopeHandler.findTLV() method is called with TAG_LOCATION_STATUS 4-A proactive command DISPLAY TEXT is sent 5-Envelope call control by sim is sent to SIM	1- Applet is triggered 2- No exception is thrown. 3- No exception is thrown.	4-91 XX

Id	Description	API/Framework Expectation	APDU Expectation
	EnvelopeHandler.getTheHandler() method is called 6-It's checked that the contents of the envelope handler is 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. It's checked that the TAG_LOCATION_STATUS is the TLV selected 7- The contents of EnvelopeHandler are compared with buffer1 using Util.arrayCompare()	5- Applet is triggered 6- No exception is thrown and the handler contains the envelope call control by SIM 7- The contents of the envelope handler shall be the same as stored in buffer 1	Proactive command Display Text is fetched The terminal Response of DISPLAY TEXT is sent to the SIM

Id	Description	API/Framework Expectation	APDU Expectation
~~13~~	~~Envelope Handler integrity checks with EVENT_ EVENT_DOWNLOAD_USER_ACTIVITY~~		
	~~1-A event download user activity envelope is sent to SIM~~	~~1- Applet is triggered~~	
	~~2-EnvelopeHandler.getTheHandler() method is called~~	~~2-No exception is thrown.~~	
	~~3-Copy the contents of the envelope handler in buffer 1 using EnvelopeHandler.copy()~~	~~3-No exception is thrown.~~	
	~~The EnvelopeHandler.findTLV() method is called with TAG_DEVICE_IDENTITIES~~		
	~~4-A proactive command DISPLAY TEXT is sent~~		~~4-91 XX~~
	~~5-Envelope call control by sim is sent to SIM~~	~~5- Applet is triggered~~	
	~~EnvelopeHandler.getTheHandler() method is called~~		
	~~6-It's checked that the contents of the envelope handler is the 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 SIM~~
	~~It's checked that the TAG_DEVICE_IDENTITIES is the TLV selected~~		
	~~7- The contents of EnvelopeHandler are compared with buffer1 using Util.arrayCompare()~~	~~7- The contents of the envelope handler shall be the same as stored in buffer 1~~	
13	Envelope Handler integrity checks with EVENT_ EVENT_DOWNLOAD_USER_ACTIVITY		
	1-A event download user activity envelope is sent to SIM	1- Applet is triggered	
	2-EnvelopeHandler.getTheHandler() method is called	2- No exception is thrown.	
	3-Copy the contents of the envelope handler in buffer 1 using EnvelopeHandler.copy()	3- No exception is thrown.	
	The EnvelopeHandler.findTLV() method is called with TAG_DEVICE_IDENTITIES		
	4-A proactive command DISPLAY TEXT is sent		4- 91 XX
	5-Envelope call control by sim is sent to SIM	5- Applet is triggered	
	EnvelopeHandler.getTheHandler() method is		

Id	Description	API/Framework Expectation	APDU Expectation
	called 6-It's checked that the contents of the envelope handler is 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. It's checked that the TAG_DEVICE_IDENTITIES is the TLV selected 7- The contents of EnvelopeHandler are compared with buffer1 using Util.arrayCompare()	6- No exception is thrown and the handler contains the envelope call control by SIM 7- The contents of the envelope handler shall be the same as stored in buffer 1	Proactive command Display Text is fetched The terminal Response of DISPLAY TEXT is sent to the SIM

Id	Description	API/Framework Expectation	APDU Expectation
~~14~~	~~**Envelope Handler integrity checks with EVENT_ EVENT_DOWNLOAD_IDLE_SCREEN_AVAILABLE**~~ ~~1-A event download idle screen available envelope is sent to SIM~~ ~~2-EnvelopeHandler.getTheHandler() method is called~~ ~~3-Copy the contents of the envelope handler in buffer 1 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 SIM~~ ~~EnvelopeHandler.getTheHandler() method is called~~ ~~6-It's checked that the contents of the envelope handler is 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.~~ ~~It's checked that the TAG_DEVICE_IDENTITIES is the TLV selected~~ ~~7- The contents of EnvelopeHandler are 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 envelope handler shall be the same as stored in buffer 1~~	~~4-91 XX~~ ~~Proactive command Display Text is fetched~~ ~~The terminal Response of DISPLAY TEXT is sent to the SIM~~
14	**Envelope Handler integrity checks with EVENT_ EVENT_DOWNLOAD_IDLE_SCREEN_AVAILABLE** 1-A event download idle screen available envelope is sent to SIM 2-EnvelopeHandler.getTheHandler() method is called 3-Copy the contents of the envelope handler in buffer 1 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 SIM	1- Applet is triggered 2- No exception is thrown. 3- No exception is thrown. 5- Applet is triggered	4- 91 XX

Id	Description	API/Framework Expectation	APDU Expectation
	EnvelopeHandler.getTheHandler() method is called 6-It's checked that the contents of the envelope handler is 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. It's checked that the TAG_DEVICE_IDENTITIES is the TLV selected 7- The contents of EnvelopeHandler are compared with buffer1 using Util.arrayCompare()	6- No exception is thrown and the handler contains the envelope call control by SIM 7- The contents of the envelope handler shall be the same as stored in buffer 1	Proactive command Display Text is fetched The terminal Response of DISPLAY TEXT is sent to the SIM
~~15~~	**~~Envelope Handler integrity checks with EVENT_ EVENT_DOWNLOAD_CARD_READER_STATUS~~** ~~1-A event download card reader status envelope is sent to SIM~~ ~~2-EnvelopeHandler.getTheHandler() method is called~~ ~~3-Copy the contents of the envelope handler in buffer 1 using EnvelopeHandler.copy()~~ ~~The EnvelopeHandler.findTLV() method is called with TAG_CARD_READER_STATUS~~ ~~4-A proactive command DISPLAY TEXT is sent~~ ~~5-Envelope call control by sim is sent to SIM~~ ~~EnvelopeHandler.getTheHandler() method is called~~ ~~6-It's checked that the contents of the envelope handler is the envelope call control using EnvelopeHandler.copy and Util.arrayCompare() methods~~ ~~The EnvelopeHandler.findTLV() method is called with TAG_DEVICE_IDENTITIES~~ ~~It's checked that the TAG_CARD_READER_STATUS is the TLV selected~~ ~~7- The contents of EnvelopeHandler are compared with buffer1 using~~	~~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 envelope~~	~~4-91 XX~~ ~~Proactive command Display Text is fetched~~ ~~The terminal Response of DISPLAY TEXT is sent to the SIM~~

Id	Description	API/Framework Expectation	APDU Expectation
	~~Util.arrayCompare()~~	~~handler shall be the same as stored in buffer 1~~	

15	**Envelope Handler integrity checks with EVENT_ EVENT_DOWNLOAD_CARD_READER_STATUS**
	
	`1-A event download card reader status envelope is sent to SIM`
	
	`2-EnvelopeHandler.getTheHandler() method is called`
	
	`3-Copy the contents of the envelope handler in buffer 1 using EnvelopeHandler.copy()`
	
	`The EnvelopeHandler.findTLV() method is called with TAG_CARD_READER_STATUS`
	
	`4-A proactive command DISPLAY TEXT is sent`
	
	`5-Envelope call control by sim is sent to SIM`
	
	`EnvelopeHandler.getTheHandler() method is called`
	
	`6-It's checked that the contents of the envelope handler is the envelope call control using EnvelopeHandler.copy and Util.arrayCompare() methods`
	
	`The EnvelopeHandler.findTLV() method is called with TAG_DEVICE_IDENTITIES`
	
	`It's checked that the TAG_CARD_READER_STATUS is the TLV selected`
	
	`7- The contents of EnvelopeHandler are 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 envelope handler shall be the same as stored in buffer 1
			4- 91 XX
			
			Proactive command Display Text is fetched
			
			The terminal Response of DISPLAY TEXT is sent to the SIM

Id	Description	API/Framework Expectation	APDU Expectation
~~16~~	~~**Envelope Handler integrity checks with UNRECOGNIZED_ENVELOPE**~~ ~~1-A unrecognized envelope is sent to SIM~~ ~~2-EnvelopeHandler.getTheHandler() method is called~~ ~~3-Copy the contents of the envelope handler in buffer 1 using EnvelopeHandler.copy()~~ ~~4-A proactive command DISPLAY TEXT is sent~~ ~~5-Envelope call control by sim is sent to SIM~~ ~~EnvelopeHandler.getTheHandler() method is called~~ ~~The EnvelopeHandler.getValueLength() is called~~ ~~6-It's checked that the contents of the envelope handler is 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.~~ ~~7- The contents of EnvelopeHandler are 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 envelope handler shall be the same as stored in buffer 1~~	~~4-91 XX~~ ~~Proactive command Display Text is fetched~~ ~~The terminal Response of DISPLAY TEXT is sent to the SIM~~
16	**Envelope Handler integrity checks with UNRECOGNIZED_ENVELOPE** 1-A unrecognized envelope is sent to SIM 2-EnvelopeHandler.getTheHandler() method is called 3-Copy the contents of the envelope handler in buffer 1 using EnvelopeHandler.copy() 4-A proactive command DISPLAY TEXT is sent 5-Envelope call control by sim is sent to SIM EnvelopeHandler.getTheHandler() method is called The EnvelopeHandler.getValueLength() is called 6-It's checked that the contents of the envelope handler is the envelope call control using EnvelopeHandler.copy() and Util.arrayCompare() methods	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	4- 91 XX

Id	Description	API/Framework Expectation	APDU Expectation
	The EnvelopeHandler.findTLV() method is called with TAG_DEVICE_IDENTITIES Call Control execution is finished. 7- The contents of EnvelopeHandler are compared with buffer1 using Util.arrayCompare()	7- The contents of the envelope handler shall be the same as stored in buffer 1	Proactive command Display Text is fetched The terminal Response of DISPLAY TEXT is sent to the SIM

Id	Description	API/Framework Expectation	APDU Expectation
17	**Envelope Handler integrity checks with EVENT_EVENT_DOWNLOAD_LANGUAGE_SELECTION**		
	`1-A event download language selection envelope is sent to SIM`	1- Applet is triggered	
	`2-EnvelopeHandler.getTheHandler() method is called`	2-No exception is thrown.	
	`3-Copy the contents of the envelope handler in buffer 1 using EnvelopeHandler.copy()`	3-No exception is thrown.	
	`The EnvelopeHandler.findTLV() method is called with TAG_EVENT_LIST`		
	`4-A proactive command DISPLAY TEXT is sent`		4-91 XX
	`5-Envelope call control by sim is sent to SIM`	5- Applet is triggered	
	`EnvelopeHandler.getTheHandler() method is called`		
	`6-It’s checked that the contents of the envelope handler is the 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– SIM
	`It’s checked that the TAG_EVENT_LIST is the TLV selected`		
	`7- The contents of EnvelopeHandler are compared with buffer1 using Util.arrayCompare()`	7- The contents of the envelope handler shall be the same as stored in buffer 1	
18	**Envelope Handler integrity checks with EVENT_EVENT_DOWNLOAD_BROWSER_TERMINATION**		
	`1-A event download browser termination envelope is sent to SIM`	1- Applet is triggered	
	`2-EnvelopeHandler.getTheHandler() method is called`	2-No exception is thrown.	
	`3-Copy the contents of the envelope handler in buffer 1 using EnvelopeHandler.copy()`	3-No exception is thrown.	
	`The EnvelopeHandler.findTLV() method is called with TAG_EVENT_LIST`		
	`4-A proactive command DISPLAY TEXT is sent`		4-91 XX
	`5-Envelope call control by sim is sent to SIM`	5- Applet is triggered	

Id	Description	API/Framework Expectation	APDU Expectation
	`EnvelopeHandler.getTheHandler() method is called`		
	`6-It’s checked that the contents of the envelope handler is the 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 SIM
	`It’s checked that the TAG_EVENT_LIST is the TLV selected`		
	`7- The contents of EnvelopeHandler are compared with buffer1 using Util.arrayCompare()`	7- The contents of the envelope handler shall be the same as stored in buffer 1	

Id	Description	API/Framework Expectation	APDU Expectation

19	**Envelope Handler integrity checks with EVENT_FORMATTED_SMS_CB**
	
	`1-An envelope SMS-CB formatted according to TS 23.048 [8] is sent to the SIM`
	
	`2-EnvelopeHandler.getTheHandler() method is called`
	
	`3-Copy the contents of the envelope handler in buffer 1 using EnvelopeHandler.copy()`
	
	`The EnvelopeHandler.findTLV() method is called with TAG_CELL_BROADCAST_PAGE`
	
	`4-A proactive command DISPLAY TEXT is sent`
	
	`5-Envelope call control by sim is sent to SIM`
	
	`EnvelopeHandler.getTheHandler() method is called`
	
	`6-It's checked that the contents of the envelope handler is 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.`
	
	`It's checked that the TAG_CELL_BROADCAST_PAGE is the TLV selected`
	
	`7- The contents of EnvelopeHandler are 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 envelope handler shall be the same as stored in buffer 1
			4-91 XX
			
			Proactive command Display Text is fetched
			
			The terminal Response of -DISPLAY TEXT is sent to the –SIM

20	**Envelope Handler integrity checks with EVENT_FORMATTED_SMS_PP_UPD**
	
	`1-Update Record EFsms instruction single and formatted is sent to the SIM`
	
	`2-EnvelopeHandler.getTheHandler() method is called`
	
	`3-Copy the contents of the envelope handler in buffer 1 using EnvelopeHandler.copy( )`
	
	`The EnvelopeHandler.findTLV() method is called with TAG_SMS_TPDU`
	
	`4-A proactive command DISPLAY TEXT is sent`
		1- Applet is triggered
		
		2- No exception is thrown.
		
		3- No exception is thrown.
			4- 91 XX

Id	Description	API/Framework Expectation	APDU Expectation
	5-Envelope call control by sim is sent to SIM EnvelopeHandler.getTheHandler() method is called 6-It's checked that the contents of the envelope handler is the 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. It's checked that the TAG_SMS_TPDU is the TLV selected 7- The contents of EnvelopeHandler are compared with buffer1 using Util.arrayCompare()	5- Applet is triggered 6- No exception is thrown and the handler contains the envelope call control by SIM 7- The contents of the envelope handler shall be the same as stored in buffer 1	Proactive command Display Text is fetched The terminal Response of DISPLAY TEXT is sent to the SIM

Id	Description	API/Framework Expectation	APDU Expectation
21	**Envelope Handler integrity checks with EVENT_UNFORMATTED_SMS_PP_UPD** 1- Update Record EFsms instruction single and unformatted is sent to the SIM 2-EnvelopeHandler.getTheHandler() method is called 3-Copy the contents of the envelope handler in buffer 1 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 SIM EnvelopeHandler.getTheHandler() method is called 6-It’s checked that the contents of the envelope handler is 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. It’s checked that the TAG_DEVICE_IDENTITIES is the TLV selected 7- The contents of EnvelopeHandler are 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 envelope handler shall be the same as stored in buffer 1.	4- 91 XX Proactive command Display Text is fetched The terminal Response of DISPLAY TEXT is sent to the SIM
22	**Check the TLV list conversion for EVENT_FORMATTED_SMS_PP_UPD** 1- An EVENT_FORMATTED_SMS_PP_UPD is sent to the SIM. 2- The findTLV(tag == device identities Tag) is called. 3- The getValueByte(offset == 0) is called. 4- The getValueByte(offset == 1) is called. 5- The findTLV(tag == address Tag) is called. 6- Check the content 7- The findTLV(tag == SMS TPDU Tag) is called.	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/Framework Expectation	APDU Expectation
	8- Check the content		
23	**Check TLV list conversion for EVENT_FORMATTED_SMS_PP_UPD** 1- The getLength() method is called	1. return the Simple TLV list length	
24	**Check TLV list conversion for EVENT_FORMATTED_SMS_PP_UPD** 1- The getEnvelopeTag() method is called	1- return *BTAG_SMS_PP_DOWNLOAD*	
25	**Check the TLV list conversion for EVENT_UNFORMATTED_SMS_PP_UPD** 1- An EVENT_UNFORMATTED_SMS_PP_UPD is sent to the SIM. 2- The findTLV(tag == device identities Tag) is called. 3- The getValueByte(offset == 0) is called. 4- The getValueByte(offset == 1) is called. 5- The findTLV(tag == address Tag) is called. 6- Check the content 7- The findTLV(tag == SMS TPDU Tag) 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.	
26	**Check TLV list conversion for EVENT_UNFORMATTED_SMS_PP_UPD** 1- The getLength() method is called	1. return the Simple TLV list length	
27	**Check TLV list conversion for EVENT_UNFORMATTED_SMS_PP_UPD** 1- The getEnvelopeTag() method is called	1- return *BTAG_SMS_PP_DOWNLOAD*	

6.3.2.3.4 Test Coverage

~~***CR Number***~~	~~***Test Case Number***~~
CRR Number	**Test Case Number**
~~CRRN1~~	~~1,2,3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19~~
CRRN1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21
~~CRRN2~~	~~1,2,3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19~~
CRRN2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21
CRRN3	22, 25
CRRN4	23, 26
CRRN5	24, 27
CRRN6	22, 25

6.3.2.4 EnvelopeResponseHandler

Test Area Reference: FWK_HIN_ERHD

6.3.2.4.1 Conformance Requirement

6.3.2.4.1.1 Normal Execution

CRRN1: At the processToolkit invocation the TLV-List is cleared.

6.3.2.4.2 Test Suite Files:

Test Script:	FWK_HIN_ERHD_1.scr
Test Applet:	FWK_HIN_ERHD_1.java
Load Script:	FWK_HIN_ERHD_1.ldr
Cleanup Script:	FWK_HIN_ERHD_1.clr
Parameter File:	FWK_HIN_ERHD_1.par

6.3.2.4.3 Test Procedure

Id	Description	API/Framework Expectation	APDU Expectation

1	`Applet1 is registered to EVENT_UNRECOGNIZED_ENVELOPE.`
	
	`1-An unrecognised envelope is sent to the SIM`
	
	`2- EnvelopeResponseHandler.getTheHandler()is called by the Applet1.`
	`3- EnvelopeResponseHandler.getLength() method is called by Applet1`
		1- Applet 1 is triggered.
		
		2- The return value shall be 0.	

6.3.2.4.4 Test Coverage

CRR Number	Test Case Number
CRRN1	1

6.3.3 Applet Triggering

6.3.3.3 EVENT_MENU_SELECTION_HELP_REQUEST

Test Area Reference: FWK_APT_EMSH~~.~~

6.3.3.3.1 Conformance Requirement

6.3.3.3.1.1 Normal Execution

~~—~~CRRN1: If an~~d~~ ENVELOPE (MENU_SELECTION_HELP_SUPPORTED) command is received for one entry supporting help, then STF shall trigger the corresponding applet.

~~6.3.3.3.1.2 Parameters error~~

~~No requirements.~~

~~6.3.3.3.1.3 Context Errors~~

~~No requirements.~~CCRN2: A toolkit applet shall be triggered by the EVENT_MENU_SELECTION_HELP_REQUEST event only if the Menu Id corresponding to the Envelope Menu Selection Help Request received by the SIM Toolkit framework was registered with the *helpSupported* value set to true.

CCRN3: If at least one menuId of a Toolkit Applet registers to EVENT_MENU_SELECTION_HELP_REQUEST, the SET UP MENU proactive command sent by the SIM Toolkit Framework shall indicate to the ME that help information is available unless all the menus entries that support help are disabled.

6.3.3.3.2 Test Suite Files

Test Script:	FWK_APT_EMSH_1.scr
Test Applet:	~~FWK_APT_EMSH_1.java~~
	~~FWK_APT_EMSH_2.java~~
~~Load Script:~~	~~FWK_APT_EMSH_1.ldr~~
~~Cleanup Script:~~	~~FWK_APT_EMSH_1.clr~~
~~Parameter File:~~	FWK_APT_EMSH_1.java
	FWK_APT_EMSH_2.java
	FWK_APT_EMSH_3.java
Load Script:	FWK_APT_EMSH_1.ldr
Cleanup Script:	FWK_APT_EMSH_1.clr
Parameter File:	FWK_APT_EMSH_1.par

6.3.3.3.3 Test Procedure

Id	Description	API/Framework Expectation	APDU Expectation
~~1~~	~~**Applet registration to EVENT_MENU_SELECTION_HELP_REQUEST and triggering**~~ ~~`ToolkitRegistry.InitMenuEntry() method is called in the constructor of applet1 and Applet2.`~~ ~~`For Applet1:`~~ ~~`MenuEntry="Applet1"`~~ ~~`Offset=0`~~ ~~`Length=menuEntry.length`~~ ~~`HelpSupported=true`~~ ~~`IconQualifier=0`~~ ~~`IconIdentifier=0`~~ ~~`For Applet2:`~~ ~~`MenuEntry="Applet2"`~~ ~~`Offset=0`~~ ~~`Length=menuEntry.length`~~ ~~`HelpSupported=true`~~ ~~`IconQualifier=0`~~ ~~`IconIdentifier=0`~~ ~~`event= EVENT_MENU_SELECTION_HELP_REQUEST`~~ ~~`1-ToolkitRegistry.isEventSet() is called in constructor.`~~ ~~`Perform SIM initialization the facility SET UP MENU and without the facilities SET EVENT LIST and POLL INTEVAL features`~~	~~1-The command must return true.~~	

Id	Description	API/Framework Expectation	APDU Expectation
	~~2-Item identifier =1~~ ~~Menu Selection Help Request envelope is sent to the SIM with the item identifier of a menu entry of applet~~ ~~3-Item identifier =2~~ ~~Menu Selection Help Request envelope is sent to the SIM with the item identifier of a menu entry of applet~~	~~2- Applet1 is triggered and applet2 is not triggered~~ ~~Applet1 finalizes~~ ~~3- Applet2 is triggered and applet1 is not triggered~~	

1	**Applet registration to EVENT_MENU_SELECTION_HELP_REQUEST and triggering**
	
	Applet1 and Applet2 are installed
	
	ToolkitRegistry.InitMenuEntry() method is called in the constructor of Applet1 and Applet2.
	
	For Applet1 (item id 1):
	MenuEntry="Applet1A"
	Offset=0
	Length=menuEntry.length
	HelpSupported=true
	IconQualifier=0
	IconIdentifier=0
	
	For Applet1 (item id 2):
	MenuEntry="Applet1B"
	Offset=0
	Length=menuEntry.length
	HelpSupported=false
	IconQualifier=0
	IconIdentifier=0
	
	event= EVENT_MENU_SELECTION_HELP_REQUEST
	1- ToolkitRegistry.isEventSet() is called in constructor.
	
	For Applet2 (item id 3):
	MenuEntry="Applet2A"
	Offset=0
	Length=menuEntry.length
	HelpSupported=true
	IconQualifier=0
	IconIdentifier=0
	
	For Applet2 (item id 4):
	MenuEntry="Applet2B"
	Offset=0
	Length=menuEntry.length
	HelpSupported=false
	IconQualifier=0
	IconIdentifier=0
	
	event= EVENT_MENU_SELECTION_HELP_REQUEST
	2- ToolkitRegistry.isEventSet() is called in constructor.
	
	Perform SIM initialization with the facility SET UP MENU and without the facilities SET EVENT LIST and POLL INTERVAL
	
	3-Item identifier = 1
	Menu Selection Help Request envelope is sent to the SIM with item identifier 1 belonging to applet1
		1- The command shall return true.
		
		2- The command shall return true.
		
		3- Applet1 is triggered and Applet2 is not triggered	

Id	Description	API/Framework Expectation	APDU Expectation
	4-Item identifier = 2 Menu Selection Help Request envelope is sent to the SIM with item identifier 2 belonging to applet1 5-Item identifier = 3 Menu Selection Help Request envelope is sent to the SIM with item identifier 3 belonging to applet2 6-Item identifier = 4 Menu Selection Help Request envelope is sent to the SIM with item identifier 4 belonging to applet2	4 Applet1 and Applet2 are not triggered 5- Applet2 is triggered and Applet1 is not triggered 6- Applet2 and Applet1 are not triggered	

2	**Applet deregistration to EVENT_MENU_SELECTION_HELP_REQUEST**
	
	Applet1 and Applet2 are deleted
	
	Applet3 is installed
	
	ToolkitRegistry.InitMenuEntry() method is called in the constructor of Applet3.
	
	For Applet3 (item id 5):
	MenuEntry="Applet3A"
	Offset=0
	Length=menuEntry.length
	HelpSupported=true
	IconQualifier=0
	IconIdentifier=0
	
	For Applet3 (item id 6):
	MenuEntry="Applet3B"
	Offset=0
	Length=menuEntry.length
	HelpSupported=true
	IconQualifier=0
	IconIdentifier=0
	
	For Applet3 (item id 7):
	MenuEntry="Applet3C"
	Offset=0
	Length=menuEntry.length
	HelpSupported=false
	IconQualifier=0
	IconIdentifier=0
	
	1. Perform SIM initialization with the facility SET UP MENU and without the facilities SET EVENT LIST and POLL INTERVAL
	
	2. Menu Selection Help Request envelope is sent to the SIM with item identifier 5 belonging to applet3
	
	3. ToolkitRegistry.disableMenuEntry() method for item id 5 is called by the Menu Selection Help Request Envelope.
		2. Applet3 is triggered by EVENT_MENU_SELECTION_HELP_REQUEST	1. The SIM shall issue a SET UP MENU proactive command with Menu Entry ID entry '05', '06' and '07', and Help supported set to true.
			
			3. The SIM shall issue a SET UP MENU proactive command with Menu Entry

Id	Description	API/Framework Expectation	APDU Expectation

	`4. Menu Selection Help Request envelope is sent to the SIM with item identifier 6 belonging to applet3`
	
	`5. ToolkitRegistry.disableMenuEntry() method for item id 6 is called by the Menu Selection Help Request Envelope.`
		4. Applet3 is triggered by EVENT_MENU_SELECTION_HELP_REQUEST	ID entry '06' and '07', and Help supported set to true.
			
			5. The SIM shall issue a SET UP MENU proactive command with Menu Entry ID entry '07', and Help supported set to false.

6.3.3.3.4 Test Coverage

CRR Number	Test Case Number
CRRN1	1
CRRN2	1
CRRN3	2

6.3.3.4 EVENT_FORMATTED_SMS_PP_ENV

Test Area Reference: FWK_APT_EFSE~~.~~

6.3.3.4.1 Conformance Requirement

6.3.3.4.1.1 Normal Execution

~~—~~CRRN1: The applet is triggered by the EVENT_FORMATTED_SMS_PP_ENV once:

- ~~–~~it has been registered to this ~~event;~~event,
- ~~– an envelope APDU carrying an SMS TPDU~~a Short Message Point to Point (Single or Concatenated) is received by Envelope APDU(s) and is formatted according to ~~GSM 03.48 [8], is received;~~ TS 23.048 [8],
- ~~–~~the toolkit applet to be triggered is registered with the corresponding TAR in the SMS ~~TPDU;~~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.

~~6.3.3.4.1.2 Parameters error~~

~~No requirements.~~

~~6.3.3.4.1.3 Context Errors~~

~~No requirements.~~

6.3.3.4.2 Test Suite Files

Test Script: FWK_APT_EFSE_1.scr

Test Applet: ~~FWK_APT_EFSE_1.java~~

~~Load Script: FWK_APT_EFSE_1.ldr~~

~~Cleanup Script: FWK_APT_EFSE_1.clr~~

~~Parameter File:~~ FWK_APT_EFSE_1.java

Load Script: FWK_APT_EFSE_1.ldr

Cleanup Script:	FWK_APT_EFSE_1.clr
Parameter File:	FWK_APT_EFSE_1.par

6.3.3.4.3 Test Procedure

Id	Description	API/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-An Envelope EVENT_FORMATTED_SMS_PP_ENV is sent to the SIM.~~	~~1- Applet is triggered~~	
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 SIM. 2- A Concatenated Short Message SMS-PP Formatted Data Download is sent to the SIM (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~~ ~~2-A formatted sms pp envelope is sent to the SIM.~~ ~~An unrecognized envelope is sent to the sim~~ ~~ToolkitRegistry.setEvent() method is called for EVENT_FORMATTED_SMS_PP_ENV~~ ~~3-An Envelope FORMATTED_SMS_PP_ENV is sent to the SIM~~	~~1- Applet is not 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 SIM.. 2- A Concatenated Short Messages SMS-PP Data Download is sent to the SIM (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 sim ToolkitRegistry.setEvent() method is called for EVENT_FORMATTED_SMS_PP_ENV 3- A Single Short Messages SMS-PP Data	1- Applet is not triggered 2- Applet is not triggered	

Id	Description	API/Framework Expectation	APDU Expectation
	Download is sent to the SIM. 4- A Concatenated Short Messages SMS-PP Data Download is sent to the SIM (composed of 2 Short Messages. The UDL for the first Short Message is 70 and for the second 70).	3- Applet is triggered 4- Applet is triggered	

6.3.3.4.4 Test Coverage

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

~~NOTE: The security checks are not relevant to the test designed in this test area; they will be checked in subclause 6.3.6.~~

Note 1: The security checks are not relevant to the test designed in this test area; they will be checked in the "Framework Security Management" section.

6.3.3.5 EVENT_UNFORMATTED_SMS_PP_ENV

Test Area Reference: FWK_APT_EUSE~~.~~

6.3.3.5.1 Conformance Requirement

6.3.3.5.1.1 Normal Execution

~~-~~CRRN1: The ~~applet is~~applets registers are triggered by the EVENT_UNFORMATTED_SMS_PP_ENV once ~~it has registered to this event and an Unformatted Envelope DataDownLoad is received if no proactive session is ongoing.~~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.

~~6.3.3.5.1.2 Parameters error~~

~~No requirements.~~

~~6.3.3.5.1.3 Context Errors~~

~~No requirements.~~

6.3.3.5.2 Test Suite Files

Test Script: FWK_APT_EUSE_1.scr

Test Applet: ~~FWK_APT_EUSE_1.java~~

~~Load Script: FWK_APT_EUSE_1.ldr~~

~~Cleanup Script: FWK_APT_EUSE_1.clr~~

~~Parameter File:~~ FWK_APT_EUSE_1.java

Load Script: FWK_APT_EUSE_1.ldr

Cleanup Script:	FWK_APT_EUSE_1.clr
Parameter File:	FWK_APT_EUSE_1.par

6.3.3.5.3 Test Procedure

Id	Description	API/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 ENVENT_FORMATTED_SMS_PP_ENV.~~ ~~1-Toolkit Registry.isEventSet() method is called for EVENT_UNFORMATTED_SMS_PP_ENV~~ ~~_~~ ~~2-An Envelope UNFORMATTED_SMS_PP_ENV is sent to the SIM.~~	~~1- The method returns true~~ ~~2- Applet is triggered~~	
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-Toolkit Registry.isEventSet() method is called for EVENT_UNFORMATTED_SMS_PP_ENV 2- A Single and Unformatted SMS-PP Data Download Envelope is sent to the SIM. 3- A Concatenated and Unformatted SMS-PP Data Download Envelope is sent to the SIM (composed of 2 Short Messages. The UDL for the first Short Message is 70 and for the second 70)	1- The method returns true 2- Applet is triggered 3- Applet is triggered	
~~2~~	~~**Applet deregistration**~~ ~~Toolkit Registry.clearEvent()method is called for EVENT_UNFORMATTED_SMS_PP_ENV~~ ~~1-An unformatted sms pp envelope is sent to the SIM.~~ ~~A formatted sms pp envelope is sent to the sim~~ ~~Toolkit Registry.setEvent() method is called for EVENT_UNFORMATTED_SMS_PP_ENV~~ ~~2-An Envelope UNFORMATTED_SMS_PP_ENV is sent to the SIM~~	~~1- Applet isn't triggered~~ ~~2- Applet is triggered~~	
2	**Applet deregistration** Toolkit Registry.clearEvent()method is called for EVENT_UNFORMATTED_SMS_PP_ENV 1- A Single and Unformatted SMS-PP Data Download Envelope is sent to the SIM.	1- Applet isn't triggered	

Id	Description	API/Framework Expectation	APDU Expectation
	2- A Concatenated and Unformatted SMS-PP Data Download Envelope is sent to the SIM (composed of 2 Short Messages. The UDL for the first Short Message is 70 and for the second 70) Applet is triggered by a EVENT_FORMATTED_SMS_PP_ENV Toolkit Registry.setEvent() method is called for EVENT_UNFORMATTED_SMS_PP_ENV 3- A Single and Unformatted SMS-PP Data Download Envelope is sent to the SIM. 4- A Concatenated and Unformatted SMS-PP Data Download Envelope is sent to the SIM (composed of 2 Short Messages. The UDL for the first Short Message is 70 and for the second 70)	2- Applet isn't triggered 3- Applet is triggered 4- Applet is triggered	

6.3.3.5.4 Test Coverage

CRR Number	Test Case Number
CRRN1	1, 2
CRRN2	2

6.3.3.22 EVENT_FIRST_COMMAND_AFTER_SELECT

Test Area Reference: FWK_APT_EFCA

6.3.3.22.1 Conformance Requirement

6.3.3.22.1.1 Normal Execution

CRRN1: The applet is triggered by the EVENT_FIRST_COMMAND_AFTER_SELECT once it has registered to this event; Upon reception of the first command received by the GSM application after it has been selected, or after the ATR if it is the default application, and before the Status Word of the processed command has been sent back by the GSM application, the toolkit framework shall trigger all the toolkit applets registered to this event.

CRRN2: The applet is not triggered by the EVENT_FIRST_COMMAND_AFTER_SELECT once it has deregistered from this event.

CRRN3: If the first command received by the GSM application is a toolkit applet triggering command (e.g. TERMINAL PROFILE), the toolkit applets registered on the EVENT_FIRST_COMMAND_AFTER_SELECT event shall be triggered first.

6.3.3.22.2 Test Suite Files

Test Script:	FWK_APT_EFCA_1.scr
Test Applet:	FWK_APT_EFCA_1.java
	FWK_APT_EFCA_2.java
	FWK_APT_EFCA_3.java
	FWK_APT_EFCA_4.java

FWK_APT_EFCA_5.java

Load Script: FWK_APT_EFCA_1.ldr

Cleanup Script: FWK_APT_EFCA_1.clr

Parameter File: FWK_APT_EFCA_1.par

6.3.3.22.3 Test Procedure

Id	Description	API/Framework Expectation	APDU Expectation
1	**Applets registration to EVENT_FIRST_COMMAND_AFTER_SELECT and triggering** Applet1 is registered to the EVENT_FIRST_COMMAND_AFTER_SELECT Applet2 is registered to the EVENT_PROFILE_DOWNLOAD. Applet3 is registered to EVENT_FORMATTED_SMS_PP_ENV. 1-Terminal Profile command is sent to the SIM. Applet1 deregisters from EVENT_FIRST_COMMAND_AFTER_SELECT. 2- Applet2 deregisters from EVENT_PROFILE_DOWNLOAD. 3-Envelope(SMS-PP-DOWNLOAD) formatted is sent to the SIM 4-Applet3 calls setEvent() on event EVENT_FIRST_COMMAND_AFTER_SELECT.	1- Applet1 is triggered by EVENT_FIRST_COMMAND_AFTER_SELECT Applet1 finalizes Applet2 is triggered by EVENT_PROFILE_DOWNLOAD Applet2 finalizes Applet3 is not triggered 3-Applet3 is triggered.	
2	**Deregistered applets are not triggered** 1-Reset then Terminal Profile command is sent to the SIM 2-Applet3 calls setEvent() on EVENT_PROFILE_DOWNLOAD.	1-Applet3 is triggered. Applet1 and Applet2 are not triggered. 2-Applet3 finalizes.	
3	**Install a 4th applet registered to EVENT_FIRST_COMMAND_AFTER_SELECT and EVENT_PROFILE_DOWNLOAD** Applet4 is installed, with the same priority level as Applet3. 1-Reset then Terminal Profile command is sent to the SIM Delete all applets.	1- Applet4 is triggered by EVENT_FIRST_COMMAND_AFTER_SELECT. Applet3 is triggered by EVENT_FIRST_COMMAND_AFTER_SELECT. Applet4 is triggered by EVENT_PROFILE DOWNLOAD. Applet3 is triggered by EVENT_PROFILE_DOWNLOAD.	
4	**Check that the applet is triggered before the first SW is sent.**		3-The SETUP MENU proactive command is

Id	Description	API/Framework Expectation	APDU Expectation

	`1-Install Applet 5.` `Applet 5 is registered with two entries in the menu entries list. Applet5 is also registered to EVENT_FIRST_COMMAND_AFTER_SELECT.`
	
	 `2-Reset and TERMINAL PROFILE.`
	
	 `3-Applet disables a menu entry.`
		2- Applet 5 is triggered	fetched. There is only one item for Applet5.

[Note: Testing the triggering of an applet upon the first command after select is not possible.]

6.3.3.22.4 Test Coverage

CR Number	Test Case Number
CRRN1	1,2,3, 4
CRRN2	3
CRRN3	1, 4

6.3.3.23 EVENT_EVENT_DOWNLOAD_DATA_AVAILABLE

Test Area Reference: FWK_APT_EDDA

6.3.3.23.1 Conformance Requirement

6.3.3.23.1.1 Normal Execution

CRRN1: For EVENT_EVENT_DOWNLOAD_DATA_AVAILABLE, the framework shall only trigger the applet registered to this event with the appropriate channel identifier.

CRRN2: The registration to the EVENT_EVENT_DOWNLOAD_DATA_AVAILABLE is effective once the toolkit applet has issued a successful OPEN CHANNEL proactive command, and valid till the first successful CLOSE CHANNEL or the end of card session.

CRRN3: When a Toolkit Applet has sent an OPEN CHANNEL proactive command and received a successful TERMINAL RESPONSE, the framework shall register the received channel identifier for the calling Toolkit Applet.

CRRN4: When a Toolkit Applet has sent a CLOSE CHANNEL proactive command and received a successful TERMINAL RESPONSE, the framework shall release the channel identifier contained in the command. A successful TERMINAL RESPONSE means that the result of the proactive command execution belongs to command performed category (i.e. General Result ='0x').

6.3.3.23.2 Test Suite Files

Test Script:	FWK_APT_EDDA_1.scr
Test Applet:	FWK_APT_EDDA_1.java
Load Script:	FWK_APT_EDDA_1.ldr
Cleanup Script:	FWK_APT_EDDA_1.clr
Parameter File:	FWK_APT_EDDA_1.par

6.3.3.23.3 Test Procedure

Id	Description	API/Framework Expectation	APDU Expectation
1	**Applet registration to EVENT_EVENT_DOWNLOAD_DATA_AVAILABLE** Applet1 is registered to Unformatted SMS PP Envelope. 1- Unformatted SMS PP envelope is sent to the SIM. 2- Applet calls setEvent() with the event EVENT_EVENT_DOWNLOAD_DATA_AVAILABLE. 3- An envelope Event Download Data Available is sent to the SIM Channel Status = 81 00 4- Unformatted SMS PP envelope is sent to the SIM. 5- Applet1 builds a proactive command OPEN CHANNEL calling ProactiveHandler.init() method. 6- send() method is called to register to this event. 8- An envelope Event Download Data Available is sent to the SIM with Channel Status = 01 00. 9- Unformatted SMS PP envelope is sent to the SIM. 10- Applet1 builds a proactive command OPEN CHANNEL calling ProactiveHandler.init() method. 11- send() method is called to register to this event.	1- Applet1 is triggered by Unformatted SMS PP envelope. 2- Applet1 finalizes. 3- Applet1 is not triggered. 4- Applet1 is triggered by Unformatted SMS PP envelope. 7- Applet1 finalizes. 8- Applet1 is not triggered. 9- Applet1 is triggered by EVENT_UNFORMATTED_SMS_PP_ENV. 12- Applet1 finalizes.	6- OPEN CHANNEL proactive command is fetched. Unsuccessful TERMINAL RESPONSE of OPEN CHANNEL is sent to the SIM. 11- OPEN CHANNEL proactive command is fetched. Successful TERMINAL RESPONSE of OPEN CHANNEL is sent to the SIM with Channel Id = 01.
2	**Applet triggering to EVENT_EVENT_DOWNLOAD_DATA_AVAILABLE** 1- An envelope Event Download Data Available is sent to the SIM Channel Status = 81 00.	1- Applet1 is triggered.	
3	**Applet deregistration to EVENT_EVENT_ DOWNLOAD_DATA_ AVAILABLE** 0- Unformatted SMS PP envelope is sent to the SIM. 1- Applet1 initializes and sends an OPEN CHANNEL proactive command. 2- Applet1 builds a CLOSE CHANNEL	0- Applet1 is triggered.	1- OPEN CHANNEL proactive command is fetched. Successful terminal response is sent, with channelId=02.

Id	Description	API/Framework Expectation	APDU Expectation
	Proactive Command calling ProactiveHandler.initCloseChannel() and ProactiveHandler.send() methods. 3- An envelope Event Download Data Available is sent to the SIM. Channel Status = 82 00 4- Applet1 builds a CLOSE CHANNEL Proactive Command calling ProactiveHandler.initCloseChannel() and ProactiveHandler.send() methods.	3- Applet1 is triggered. 5- Applet1 finalizes.	2- CLOSE CHANNEL proactive command is fetched. Unsuccessful TERMINAL RESPONSE of CLOSE CHANNEL is sent to the SIM. 4- CLOSE CHANNEL proactive command is fetched. Successful TERMINAL RESPONSE of CLOSE CHANNEL is sent to the SIM with Channel Id = 02.
4	**Applet triggering to EVENT_EVENT_DOWNLOAD_DATA_AVAILABLE** 1- An envelope Event Download Data Available is sent to the SIM Channel Status = 82 00.	1- Applet1 is not triggered.	
5	**Applet1 not triggered after a reset** 0- Applet1 is triggered by an unformatted SMS PP Envelope 1- Applet1 builds a proactive command OPEN CHANNEL calling ProactiveHandler.init() method. 2- send() method is called to register to this event. 3- isEventSet() method is called. 4- Reset the card. 5- An envelope Event Download Data Available is sent to the SIM Channel Status = 82 00.	3- returns true. 5- Applet1 is not triggered.	1- OPEN CHANNEL proactive command is fetched. 2- Successful TERMINAL RESPONSE of OPEN CHANNEL is sent to the SIM with Channel Id = 02.

6.3.3.23.4 Test Coverage

CR Number	Test Case Number
CRRN1	2
CRRN2	1, 4, 5
CRRN3	1
CRRN4	3

6.3.3.24 EVENT_EVENT_DOWNLOAD_CHANNEL_STATUS

Test Area Reference: FWK_APT_EDCS

6.3.3.24.1 Conformance Requirement

6.3.3.24.1.1 Normal Execution

CRRN1: For EVENT_EVENT_DOWNLOAD_CHANNEL_STATUS, the framework shall only trigger the applet registered to this event with the appropriate channel identifier.

CRRN2: The registration to the EVENT_EVENT_DOWNLOAD_CHANNEL_STATUS is effective once the toolkit applet has issued a successful OPEN CHANNEL proactive command, and valid till the first successful CLOSE CHANNEL or the end of the card session.

CRRN3: When a Toolkit Applet has sent an OPEN CHANNEL proactive command and received a successful TERMINAL RESPONSE, the framework shall register the received channel identifier for the calling Toolkit Applet.

CRRN4: When a Toolkit Applet has sent a CLOSE CHANNEL proactive command and received a successful TERMINAL RESPONSE, the framework shall release the channel identifier contained in the command. A successful TERMINAL RESPONSE means that the result of the proactive command execution belongs to command performed category (i.e. General Result ='0x').

6.3.3.24.2 Test Suite Files

Test Script:	FWK_APT_EDCS_1.scr
Test Applet:	FWK_APT_EDCS_1.java
Load Script:	FWK_APT_EDCS_1.ldr
Cleanup Script:	FWK_APT_EDCS_1.clr
Parameter File:	FWK_APT_EDCS_1.par

6.3.3.24.3 Test Procedure

Id	Description	API/Framework Expectation	APDU Expectation

1	**Applet registration to EVENT_EVENT_DOWNLOAD_CHANNEL_STATUS**
	
	`Applet1 is registered to Unformatted SMS PP Envelope.`
	
	`1-Unformatted SMS PP envelope is sent to the SIM.`
	
	`2-The applet calls setEvent() with EVENT_EVENT_DOWNLOAD_CHANNEL_STATUS.`
	
	`3- An envelope Event Download Channel Status is sent to the SIM. Channel Status = 81 00`
	
	`4-Unformatted SMS PP envelope is sent to the SIM.`
	
	`5- Applet1 builds a proactive command OPEN CHANNEL calling ProactiveHandler.init() method.`
	
	`6- send() method is called to register to this event.`
	
	`8- An envelope Event Download Data Available is sent to the SIM with Channel Status = 01 00.`
	
	`9- Unformatted SMS PP envelope is sent to the SIM.`
	
	`10- Applet1 builds a proactive command OPEN CHANNEL calling ProactiveHandler.init() method.`
	
	`11- send() method is called to register to this event a second time.`
		1- Applet1 is triggered by Unformatted SMS PP envelope
		
		2- Applet1 finalizes.
		
		3- Applet1 is not triggered.
		
		4- Applet1 is triggered by Unformatted SMS PP envelope.
		
		7- Applet finalizes.
		
		8- Applet1 is not triggered.
		
		9- Applet1 is triggered by EVENT_UNFORMATTED_SMS_PP_ENV.
		
		12- Applet1 finalizes.
			6- OPEN CHANNEL proactive command is fetched. Unsuccessful TERMINAL RESPONSE of OPEN CHANNEL is sent to the SIM.
			
			11- OPEN CHANNEL proactive command is fetched. Successful TERMINAL RESPONSE of OPEN CHANNEL is sent to the SIM with Channel Id = 01.

2	**Applet triggering to EVENT_EVENT_DOWNLOAD_CHANNEL_STATUS** `1- An envelope Event Download Channel Status is sent to the SIM. Channel Status = 81 00`	1- Applet1 is triggered.	

3	**Applet deregistration to EVENT_EVENT_DOWNLOAD_CHANNEL STATUS**
	
	`0- Unformatted SMS PP envelope is sent to the SIM.`
	
	`1-Applet1 initializes and sends an OPEN CHANNEL proactive command.`
	
	`2- Applet1 builds a CLOSE CHANNEL Proactive Command calling ProactiveHandler.initCloseChannel() and ProactiveHandler.send() methods.`
		0- Applet1 is triggered.
		
		3- The applet is triggered.
			OPEN CHANNEL proactive command is fetched. Successful terminal response is sent, with channelId=02.
			
			2-CLOSE CHANNEL proactive command is fetched. Unsuccessful TERMINAL

Id	Description	API/Framework Expectation	APDU Expectation
	3-An envelope Event Download Channel Status is sent to the SIM. Channel Status = 82 00 4- Applet1 builds a Close Channel Proactive Command calling ProactiveHandler.initCloseChannel() and ProactiveHandler.send() methods.	5- Applet1 finalizes.	RESPONSE of CLOSE CHANNEL is sent to the SIM. 4- CLOSE CHANNEL proactive command is fetched. Successful TERMINAL RESPONSE of CLOSE CHANNEL is sent to the SIM with Channel Id = 02.
4	**Applet triggering to EVENT_EVENT_DOWNLOAD_CHANNEL STATUS** 1- An envelope Event Download Channel Status is sent to the SIM. Channel Status = 82 00	Applet1 is not triggered.	
5	**Applet1 not triggered after a reset** 0- Applet1 is triggered by an unformatted SMS PP Envelope. 1- Applet1 builds a proactive command OPEN CHANNEL calling ProactiveHandler.init() method. 2- send() method is called to register to this event. 3- isEventSet() method is called. 4- Reset the card. 5- An envelope Event Download Data Available is sent to the SIM Channel Status = 82 00.	3- returns true. 5- Applet1 is not triggered.	1- OPEN CHANNEL proactive command is fetched. 2- Successful TERMINAL RESPONSE of OPEN CHANNEL is sent to the SIM with Channel Id = 02.

6.3.3.24.4 Test Coverage

CR Number	Test Case Number
CRRN1	2
CRRN2	1, 4, 5
CRRN3	1
CRRN4	3

6.3.3.25 EVENT_FORMATTED_SMS_PP_UPD

Test Area Reference: FWK_APT_EFSU

6.3.3.25.1 Conformance Requirement

6.3.3.25.1.1 Normal Execution

CRRN1: The applet is triggered by the EVENT_FORMATTED_SMS_PP_UPD once:

- it has been registered to this event,
- a Short Message Point to Point (Single or Concatenated) is received by Update Record EFsms APDU(s) and is formatted according to TS 23.048 [8],
- 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.

6.3.3.25.2 Test Suite Files

Test Script:	FWK_APT_EFSU_1.scr
Test Applet:	FWK_APT_EFSU_1.java
Load Script:	FWK_APT_EFSU_1.ldr
Cleanup Script:	FWK_APT_EFSU_1.clr
Parameter File:	FWK_APT_EFSU_1.par

6.3.3.25.3 Test Procedure

6.3.3.25.4 Test Coverage

Id	Description	API/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. Toolkit Registry.isEventSet() method is called for EVENT_FORMATTED_SMS_PP_UPD 2. Short Message Point to Point Single and Formatted is received by Update Record EFsms APDU. 3. Short Message Point to Point Concatenated Formatted is received by Update Record EFsms APDU(s) (The Concatenated Message is composed of 2 Short Messages. The UDL for the first Short Message is 70 and for the second 70).	1- The method returns true. 2- Applet is triggered. 3- Applet is triggered on reception of the last concatenated SMS	
2	**Applet deregistration** ToolkitRegistry.clearEvent() method is called for EVENT_FORMATTED_SMS_PP_UPD 1. Short Message Point to Point Single and Formatted is received by Update Record EFsms APDU. 2. Short Message Point to Point Concatenated and Formatted is received by Update Record EFsms APDU(s). (The Concatenated Message is composed of 2 Short Messages. The UDL for the first Short Message is 70 and for the second 70). An unrecognized envelope is sent to the sim ToolkitRegistry.setEvent() method is called for EVENT_FORMATTED_SMS_PP_UPD 3. Short Message Point to Point Single and Formatted is received by Update Record EFsms APDU. 4. Short Message Point to Point Concatenated Formatted is received by Update Record EFsms APDU(s). (The Concatenated Message is composed of 2 Short Messages. The UDL for the first Short Message is 70 and for the second 70).	1- Applet is not triggered 2- Applet is not triggered 3- Applet is triggered 4- Applet is triggered on reception of the last concatenated SMS.	

CRR Number	Test Case Number
CRRN1 (See note1)	1,2
CRRN2	2

Note 1: The security checks are not relevant to the test designed in this test area; they will be checked in the "Framework Security Management" section.

6.3.3.26 EVENT_UNFORMATTED_SMS_PP_UPD

Test Area Reference: FWK_APT_EUSU

6.3.3.26.1 Conformance Requirement

6.3.3.26.1.1 Normal Execution

CRRN1: The applets registers are triggered by the EVENT_UNFORMATTED_SMS_PP_UPD once a Short Message Point to Point (Single or Concatenated) is received by Update Record EFsms APDU(s) and is unformatted.

CRRN2: The applets are not triggered by the EVENT_UNFORMATTED_SMS_PP_UPD once it has deregistered from this event.

6.3.3.26.2 Test Suite Files

Test Script:	FWK_APT_EUSU_1.scr
Test Applet:	FWK_APT_EUSU_1.java
Load Script:	FWK_APT_EUSU_1.ldr
Cleanup Script:	FWK_APT_EUSU_1.clr
Parameter File:	FWK_APT_EUSU_1.par

6.3.3.26.3 Test Procedure

Id	Description	API/Framework Expectation	APDU Expectation
1	**Applet registration to EVENT UNFORMATTED_SMS_PP_UPD and triggering** Applet is registered to EVENT_UNFORMATTED_SMS_PP_UPD and EVENT_UNRECOGNIZED_ENVELOPE 1. ToolkitRegistry.isEventSet() method is called for EVENT_UNFORMATTED_SMS_PP_UPD 2. Short Message Point to Point Single and Unformatted is received by Update Record EFsms APDU 3. Short Message Point to Point Concatenated and Unformatted is received by Update Record EFsms APDU (The Concatenated Message is composed of 2 Short Messages. The UDL for the first Short Message is 70 and for the second 70).	1- Applet is not triggered 2- Applet is triggered. 3- Applet is triggered on reception of the last concatenated SMS.	
2	**Applet deregistration** ToolkitRegistry.clearEvent() method is called for EVENT_UNFORMATTED_SMS_PP_UPD 1. Short Message Point to Point Single and Unformatted is received by Update Record EFsms APDU 2. Short Message Point to Point Concatenated and Unformatted is received by Update Record EFsms APDU(s) (The Concatenated Message is composed of 2 Short Messages. The UDL for the first Short Message is 70 and for the second 70). An unrecognized envelope is sent to the sim ToolkitRegistry.setEvent() method is called for EVENT_UNFORMATTED_SMS_PP_UPD 3. Short Message Point to Point Single and Unformatted is received by Update Record EFsms APDU 4. Short Message Point to Point Concatenated and Unformatted is received by Update Record EFsms APDU(s) (The Concatenated Message is composed of 2 Short Messages. The UDL for the first Short Message is 70 and for the second 70).	- Applet is not triggered 2- Applet is not triggered. 3- Applet is triggered 4- Applet is triggered on reception of the last concatenated SMS	

6.3.3.26.4 Test Coverage

CRR Number	Test Case Number
CRRN1	1,2
CRRN2	2

6.3.4 Proactive Command Sending by the STF

6.3.4.3 Proactive Command Control

Test Area Reference: FWK_PCS_PCCO

6.3.4.1.1 Conformance Requirements

6.3.4.1.1.1 Normal Execution

CRRN1: The SIM Toolkit Framework shall prevent the toolkit applet to issue the following proactive commands: SET UP MENU, SET UP EVENT LIST, POLL INTERVAL, POLLING OFF. If an applet attempts to issue such a command, the SIM Toolkit Framework shall throw an exception.

CRRN2: The SIM Toolkit Framework shall prevent a toolkit applet to issue a TIMER MANAGEMENT proactive command using a timer identifier, which is not allocated to it. If an applet attempts to issue such a command, the SIM Toolkit Framework shall throw an exception.

CRRN3: The SIM Toolkit Framework shall prevent a toolkit applet to issue a SEND DATA, RECEIVE DATA and CLOSE CHANNEL proactive commands using a channel identifier, which is not allocated to it. If an applet attempts to issue such a command the SIM Toolkit Framework shall throw an exception.

CRRN4: The SIM Toolkit Framework shall prevent a toolkit applet to issue an OPEN CHANNEL proactive command if it exceeds the maximum number of channel allocated to this applet. If an applet attempts to issue such a command the SIM Toolkit Framework shall throw an exception.

6.3.4.1.2 Test Suite Files

Test Script:	FWK_PCS_PCCO_1.scr
Test Applet:	FWK_PCS_PCCO_1.java
	FWK_PCS_PCCO_2.java
	FWK_PCS_PCCO_3.java
Load Script:	FWK_PCS_PCCO_1.ldr
Cleanup Script:	FWK_PCS_PCCO_1.clr
Parameter File:	FWK_PCS_PCCO_1.par

6.3.4.1.3 Test Procedure

Id	Description	API/Framework Expectation	APDU Expectation
0	**Applets installation** Applet1 is installed with 4 timers maximum, 0 channel maximum and 1 menu. Applet2 is installed with 8 timers maximum, 3 channels maximum. Applet3 is installed with 1 channel maximum.		
1	**STK Proactive Commands** 1- Send a formatted envelope with the TAR of Applet1 2- Applet1 builds and sends a SET UP MENU proactive command	1- Applet1 is triggered 2- COMMAND_NOT_ALLOWED toolkit exception is thrown	1- 90 00 (no proactive command is sent)

Id	Description	API/Framework Expectation	APDU Expectation
	3- Applet1 builds and sends a SET UP EVENT LIST proactive command 4- Applet1 builds and sends a POLL INTERVAL proactive command 5- Applet1 builds and sends a POLLING OFF proactive command	3- COMMAND_NOT_ALLOWED toolkit exception is thrown 4- COMMAND_NOT_ALLOWED toolkit exception is thrown 5- COMMAND_NOT_ALLOWED toolkit exception is thrown	
2	**TIMER MANAGEMENT Proactive command** 1- Send a formatted envelope with the TAR of Applet2 2- Applet2 allocates 8 timers by calling allocateTimer() method and release the 3 timers from id 1 to 3. 3- Send a formatted envelope with the TAR of Applet1 4- Applet1 allocates 3 timers (Id 1 to 3) by calling allocateTimer() method 3 times 5- Send a formatted envelope with the TAR of Applet2 6- Applet2 releases timers of Id 4 to 7 7- Send a formatted envelope with the TAR of Applet1 8- For each of the 3 timers allocated by Applet1 (Id 1to 3) a TIMER MANAGEMENT proactive session is performed 9- For other timers (Id 4 to 8), Applet1 builds and sends a TIMER MANAGEMENT proactive command	1- Applet2 is triggered 2- No exception is thrown 3- Applet1 is triggered 4- No exception is thrown 5- Applet2 is triggered 6- No exception is thrown 7- Applet1 is triggered 8- No exception is thrown 9- COMMAND_NOT_ALLOWED toolkit exception is thrown	8- 3 TIMER MANAGEMENT proactive commands are fetched 9- The Status word of the last previous Terminal Response is 90 00 (no more proactive command is sent)
3	**No Channel allowed** 1- Send a formatted envelope with the TAR of Applet1 2- Applet1 builds and sends a CSD OPEN CHANNEL proactive command 3- Applet1 builds and sends a GPRS OPEN CHANNEL proactive command 4Applet1 builds and sends a SEND DATA proactive command 5- Applet1 builds and sends a RECEIVE DATA proactive command 6- Applet1 builds and sends a CLOSE CHANNEL proactive command	1- Applet1 is triggered 2- COMMAND_NOT_ALLOWED toolkit exception is thrown 3- COMMAND_NOT_ALLOWED toolkit exception is thrown 4- COMMAND_NOT_ALLOWED toolkit exception is thrown 5- COMMAND_NOT_ALLOWED toolkit exception is thrown 6- COMMAND_NOT_ALLOWED toolkit exception is thrown	1- 90 00 (no proactive command is sent)
4	**4 Channels allowed** 1- Send a formatted envelope with the TAR of Applet3 2- Applet3 builds and sends a CSD OPEN CHANNEL proactive command 3- Send a Fetch and Terminal Response OK on channel 7 4- Send a formatted envelope with the TAR of Applet2 5- Applet2 builds and sends a CSD OPEN CHANNEL proactive command 6- Send a Fetch and Terminal Response OK on channel 1	1- Applet3 is triggered 2- No exception is thrown 4- Applet2 is triggered 5- No exception is thrown	2- 91 1C 3- OPEN CHANNEL proactive 5- 91 1C 6- OPEN CHANNEL proactive command is fetched

Id	Description	API/Framework Expectation	APDU Expectation
	7- Applet2 builds and sends a GPRS OPEN CHANNEL proactive command 8- Send Fetch and Terminal Response OK on channel 2 9- For each channel id from 3 to 7, Applet2 builds and sends a SEND DATA proactive command 10- For each channel id from 3 to 7, Applet2 builds and sends a RECEIVE DATA proactive command 11- For each channel id from 3 to 7, Applet2 builds and sends a CLOSE CHANNEL proactive command 12- Applet2 builds and sends a CSD OPEN CHANNEL proactive command 13- Fetch and Terminal Response OK on channel 3 14- Applet2 builds and sends an OPEN CHANNEL proactive command	7- No exception is thrown 9- COMMAND_NOT_ALLOWED toolkit exception is thrown 10- COMMAND_NOT_ALLOWED toolkit exception is thrown 11- COMMAND_NOT_ALLOWED toolkit exception is thrown 12- No exception is thrown 14- COMMAND_NOT_ALLOWED toolkit exception is thrown	7- 91 17 8- OPEN CHANNEL proactive command is fetched, SW = 91 1C on the Terminal Response 13- OPEN CHANNEL proactive command is fetched 14- 90 00 expected to the previous Terminal Response (no proactive command is sent)

6.3.4.1.4 Test Coverage

CRR number	Test case number
N1	1
N2	2
N3	3,4
N4	3,4

6.3.6 Framework Security Management

Security Parameters

The table that follows contains the security parameters that shall be used when the ~~03.48~~ TS 23.048 [8] security is required in the test cases developed in the current section.

Parameter	Value in hexadecimal
KIC	11
KID	11
CNTR	00 00 00 00 01
Key for ciphering	01 41 42 7F DA E8 91 A7
Key for RC/CC/DS	01 23 45 67 89 AB CD EF

If a parameter is not listed explicitly in the above table, the default values of section 4.7.3.1 apply.

6.3.6.1 Input Data

Test Area Reference: FWK_FWS_INDA

6.3.6.1.1 Conformance Requirements

6.3.6.1.1.1 Normal Execution

CRRN1: If the SIM receives an envelope APDU containing an SMS_PP_DATADOWNLOAD BER TLV formatted according to TS 23.048 [8], the SIM Toolkit Framework shall verify the security of the SMS TPDU.

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

CRRN3: If the SIM receives an envelope APDU containing an SMS_CB_DATADOWNLOAD formatted according to TS 23.048 [8], the SIM Toolkit Framework shall verify the security of the cell broadcast page.

CRRN4: If the SIM receives an Update Record EFsms instruction formatted according to TS 23.048[8], the SIM Toolkit Framework shall verify the security of the SMS.

CRRN5: The STF shall provide the input data deciphered.

6.3.6.1.2 Test Area Files

Test Script:	FWK_FWS_INDA_1.scr
Test Applet:	FWK_FWS_INDA_1.java
	FWK_FWS_INDA_2.java
	FWK_FWS_INDA_3.java
	FWK_FWS_INDA_4.java
	FWK_FWS_INDA_5.java
	FWK_FWS_INDA_6.java
Load Script:	FWK_FWS_INDA_1.ldr
Cleanup Script:	FWK_FWS_INDA_1.clr
Parameter File:	FWK_FWS_INDA_1.par

6.3.6.1.3 Test Procedure

Id	Description	API/Framework Expectation	APDU Expectation
~~1~~	**~~Framework checks the Cryptographic checksum and deciphers the data~~** ~~Applet1 is loaded and installed~~ ~~1 Envelope(SMS-PP) formatted is sent to the SIM with this features:~~ ~~Ciphering;~~ ~~Cryptographic checksum;~~ ~~No proof of receipt;~~ ~~Data = 01~~	~~The applet is triggered.~~	

Id	Description	API/Framework Expectation	APDU Expectation

1	**Framework checks the Cryptographic checksum and deciphers the data**
	
	`Applet1 is loaded and installed`
	
	`1-Envelope(SMS-PP) single and formatted is sent to the SIM with this features:`
	`Ciphering;`
	`Cryptographic checksum;`
	`No proof of receipt;`
	`TAR of Applet 1;`
	`Data = 01`
	
	`2- Short Message concatenated and formatted is sent to the SIM by an Envelope (SMS PP)with these features:`
	`Ciphering;`
	`Cryptographic checksum;`
	`No proof of receipt;`
	`TAR of Applet 1;`
	`Data length is 150.`
		1- The applet 1 is triggered and the value integrity is checked.
		
		2- The applet 1 is triggered and the value integrity is checked
			1- The SIM answers to the Envelope with status words 9000
			
			2- The SIM answers to the Envelope with status words 9000

~~2~~	**~~Triggering two different applets with different security~~** ~~`Applet2 is installed`~~ ~~`1-Envelope(SMS-PP) formatted is sent to the SIM with this features:`~~ ~~`Ciphering;`~~ ~~`Cryptographic checksum;`~~ ~~`No proof of receipt;`~~ ~~`TAR of Applet 1`~~ ~~`Data = 02`~~ ~~`2-Envelope(SMS-PP) formatted is sent to the SIM with this features:`~~ ~~`No ciphering;`~~ ~~`Cryptographic checksum;`~~ ~~`No proof of receipt;`~~ ~~`TAR of Applet 2`~~ ~~`Data = 03`~~	~~1- Applet1 is triggered~~ ~~2- Applet2 is triggered~~	~~1- The SIM answers to the Envelope with status words 9000~~ ~~2- The SIM answers to the Envelope with status words 9000~~
~~3~~	**~~Envelope(SMS-PP) formatted with wrong cryptographic checksum~~** ~~`No ciphering;`~~ ~~`Wrong Cryptographic checksum;`~~ ~~`No proof of receipt;`~~ ~~`TAR of Applet 1`~~ ~~`Data = 04`~~	~~No applet is triggered~~	~~1- The SIM answers to the Envelope with status words 9000~~

Id	Description	API/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 SIM with this features: Ciphering; Cryptographic checksum; No proof of receipt; TAR of Applet 1 Data = 03 2- Short Message concatenated and formatted is sent to the SIM by an Envelope (SMS PP)with these features: Ciphering; Cryptographic checksum; No proof of receipt; TAR of Applet 1 Data length = 150 3-Envelope(SMS-PP) single and formatted is sent to the SIM with this features: No ciphering; No cryptographic checksum; No proof of receipt; TAR of Applet 2 Data = 05 4- Short Message concatenated and formatted is sent to the SIM by an Envelope (SMS PP)with these features:: No ciphering; No cryptographic checksum; No proof of receipt; TAR of Applet 2 Data length = 150.	1- Applet 1 is triggered and the value integrity is checked 2- Applet 1 is triggered and the value integrity is checked 3- Applet 2 is triggered and the value integrity is checked 4- Applet 2 is triggered and the value integrity is checked	1- The SIM answers to the Envelope with status words 9000 2- The SIM answers to the Envelope with status words 9000 3- The SIM answers to the Envelope with status words 9000 4- The SIM 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 SIM with this features: No ciphering; Wrong cryptographic checksum; No proof of receipt; TAR of Applet 1 Data = 07 2- Short Message concatenated and formatted is sent to the SIM by an Envelope (SMS PP)with these features: No ciphering; Wrong cryptographic checksum; No proof of receipt; TAR of Applet 1 Data length = 150	1- No applet is triggered. 2- No applet is triggered.	1- The SIM answers to the Envelope with status words 9000

Id	Description	API/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 SIM with this features:~~ ~~Ciphering;~~ ~~Cryptographic checksum;~~ ~~No proof of receipt;~~ ~~Data = 01~~	~~1- Applet3 is triggered.~~	~~1- The SIM answers to the Envelope with status words 9000~~
4	**Framework checks the Cryptographic checksum and deciphers the data** Applet3 is loaded and installed 1-Envelope(SMS-CB) formatted is sent to the SIM with this features: Ciphering; Cryptographic checksum; No proof of receipt; Data = 01	1- Applet3 is triggered and the value integrity is checked	1- The SIM 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 SIM with this features:~~ ~~Ciphering;~~ ~~Cryptographic checksum;~~ ~~No proof of receipt;~~ ~~TAR of Applet 3~~ ~~Data = 02~~ ~~2-Envelope(SMS-CB) formatted is sent to the SIM with this features:~~ ~~No ciphering;~~ ~~Cryptographic checksum;~~ ~~No proof of receipt;~~ ~~TAR of Applet 4~~ ~~Data = 03~~	~~1- Applet3 is triggered~~ ~~2- Applet4 is triggered~~	~~1- The SIM answers to the Envelope with status words 9000~~ ~~2- The SIM answers to the Envelope with status words 9000~~

Id	Description	API/Framework Expectation	APDU Expectation
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 SIM with this features: Ciphering; Cryptographic checksum; No proof of receipt; TAR of Applet 3 Data = 02 2-Envelope(SMS-CB) formatted is sent to the SIM with this features: No ciphering; No cryptographic checksum; No proof of receipt; TAR of Applet 4 Data = 03	1- Applet3 is triggered and the value integrity is checked 2- Applet4 is triggered and the value integrity is checked	1- The SIM answers to the Envelope with status words 9000 2- The SIM 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 Applet 3~~ ~~Data = 04~~	~~No applet is triggered~~	~~The SIM 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 Applet 3
	Data = 04
		No applet is triggered	1- The SIM answers to the Envelope with status words 9000

7	**Framework checks the Cryptographic checksum and deciphers the data**
	
	Applet5 is installed
	
	1- Short Message single and formatted is sent to the SIM by Update Record EFsms instruction with these features:
	Ciphering;
	Cryptographic checksum;
	No proof of receipt;
	TAR of Applet5;
	Data = 01
	
	2- Short Message concatenated and formatted is sent to the SIM by Update Record EFsms instruction with these features:
	Ciphering;
	Cryptographic checksum;
	No proof of receipt;
	TAR of Applet5;
	Data length = 150.
		1- The applet5 is triggered and the value integrity is checked.
		
		2- The applet5 is triggered and the value integrity is checked
			1- The SIM answers to the Update Record EFsms instruction with status words 9000
			
			2- The SIM answers to the Update Record EFsms instruction with status words 9000

8	**Triggering two different applets with different security** Applet6 is installed 1- Short Message single and formatted is sent to the SIM by Update Record EFsms instruction with these features: Ciphering; Cryptographic checksum;	1- Applet5 is triggered and the value integrity is checked.	1- The SIM answers to the Update Record EFsms instruction with status words 9000

Id	Description	API/Framework Expectation	APDU Expectation
	No proof of receipt; TAR of Applet5 Data = 03 2- Short Message concatenated and formatted is sent to the SIM by Update Record EFsms instruction with these features: Ciphering; Cryptographic checksum; No proof of receipt; TAR of Applet5 Data length = 150. 3- Short Message single and formatted is sent to the SIM by Update Record EFsms instruction with these features: No ciphering; No cryptographic checksum; No proof of receipt; TAR of Applet6; Data = 05 4- Short Message concatenated and formatted is sent to the SIM by Update Record EFsms instruction with these features: No ciphering; No cryptographic checksum; No proof of receipt; TAR of Applet6; Data length = 150.	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.	2- The SIM answers to the Update Record EFsms instruction with status words 9000 3- The SIM answers to the Update Record EFsms instruction with status words 9000 4- The SIM answers to the Update Record EFsms instruction with status words 9000
9	**Update Record EFsms instruction formatted with wrong cryptographic checksum** 1- Short Message single and formatted is sent to the SIM by Update Record EFsms instruction with these features:No ciphering; Wrong Cryptographic checksum; No proof of receipt; TAR of Applet5 Data = 07 2- Short Message concatenated and formatted is sent to the SIM by Update Record EFsms instruction with these features: No ciphering; Wrong Cryptographic checksum; No proof of receipt; TAR of Applet5 Data length = 150	1- No applet is triggered. 2- No applet is triggered.	1- The SIM answers to the Update Record EFsms instruction with status words 9000 2- The SIM answers to the Update Record EFsms instruction with status words 9000

6.3.6.1.4 Test Coverage

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

6.3.7 Envelope Response Posting

6.3.7.4 EVENT_FORMATTED_SMS_PP_ENV

Test Area Reference: FWK_ERP_EFSE

6.3.7.4.1 Conformance Requirement

6.3.7.4.1.1 Normal Execution

CRRN1: If PoR is required a SMS-DELIVER REPORT is sent by the SIM, when the post() or the postAsBERTLV() method is invoked and if bit 6 of the second octet of SPI is set to 0.

CRRN2: If PoR is required a SMS-SUBMIT is sent by the SIM, when the post() or the postAsBERTLV() method is invoked and if bit 6 of the second octet of SPI is set to 1. In this case the statusType method parameter is meaningless. The SIM Toolkit Framework shall build and issue a Send Short Message proactive command as defined in TS 11.14 [4].

6.3.7.4.2 Test Suite Files

Test Script:	FWK_ERP_EFSE_1.scr
Test Applet:	FWK_ERP_EFSE_1.java
	FWK_ERP_EFSE _2.java
Load Script:	FWK_ERP_EFSE _1.ldr
Cleanup Script:	FWK_ERP_EFSE _1.clr
Parameter File:	FWK_ERP_EFSE _1.par

6.3.7.4.3 Test Procedure

Id	Description	API/Framework Expectation	APDU Expectation
1	**SMS DELIVER REPORT** 1- A formatted sms pp envelope with SMS Deliver Report required is sent to the SIM with bit 6 of SPI2 set to 0. 2- EnvelopeResponseHandler.getTheHandler() method is called by Applet1 3- Applet1 builds the answer and calls the post() method with StatusType=SW1_RP_ACK 4- A formatted sms pp envelope with SMS Deliver Report required is sent to the SIM with bit 6 of SPI2 set to 0. 5- EnvelopeResponseHandler.getTheHandler() method is called by Applet1 6- Applet1 builds the answer and calls the postAsBERTLV() method with StatusType=SW1_RP_ACK	1- Applet1 is triggered 2- No exception is thrown. Applet1 finalizes 4- Applet1 is triggered 5- No exception is thrown. Applet1 finalizes	3- ME receives 9FXX and checks the response 5- ME receives 9FXX and checks the response
2	**SMS-SUBMIT** 1- A formatted sms pp envelope with SMS Submit required is sent to the SIM with bit 6 of SPI2 set to 1. 2- EnvelopeResponseHandler.getTheHandler() method is called by Applet1 3-Applet1 builds the answer and calls the post() method with StatusType=SW1_RP_ACK 4- A formatted sms pp envelope with SMS Submit required is sent to the SIM with bit 6 of SPI2 set to 1. 5- EnvelopeResponseHandler.getTheHandler() method is called by Applet1	1- Applet1 is triggered 2- No exception is thrown. Applet1 finalizes 4- Applet1 is triggered 5- No exception is thrown	3- ME receives a Send Short Message proactive command.

Id	Description	API/Framework Expectation	APDU Expectation
	6- Applet1 builds the answer and calls the post() method with StatusType=SW1_RP_ERROR	Applet1 finalizes	6- ME receives a Send Short Message proactive command.
	7- A formatted sms pp envelope with SMS Submit required is sent to the SIM with bit 6 of SPI2 set to 1.	7- Applet1 is triggered	
	8- EnvelopeResponseHandler.getTheHandler() method is called by Applet1	8- No exception is thrown.	
	9.-Applet1 builds the answer and calls the postAsBERTLV() method with StatusType=SW1_RP_ACK	Applet1 finalizes	9- ME receives a Send Short Message proactive command.
	10- A formatted sms pp envelope with SMS Submit required is sent to the SIM with bit 6 of SPI2 set to 1.	10- Applet1 is triggered	
	11- EnvelopeResponseHandler.getTheHandler() method is called by Applet1	11- No exception is thrown.	
	12- Applet1 builds the answer and calls the postAsBERTLV () method with StatusType=SW1_RP_ERROR		12- ME receives a Send Short Message proactive command.

6.3.7.4.4 Test Coverage

CRR Number	Test Case Number
CRRN1	1
CRRN2	2

6.3.8 Toolkit Installation

6.3.8.1 Timers Allocation

Test Area Reference: FWK_TIN_TMAL~~.~~

6.3.8.1.1 Conformance Requirements

6.3.8.1.1.1 Normal execution

CRRN1: One toolkit applet can register to several timers, but a timer can only be allocated to one toolkit applet.

6.3.8.1.1.2 ~~Parameters error~~

~~No requirements.~~

~~6.3.8.1.1.3~~ Context errors

~~-~~CRRC1 : Allocated timers shall not exceed the maximum number of timers allowed for this applet instance defined during installation.

~~-~~CRRC2 : The total number of timers allocated for all the applets shall not exceed 8. If the maximum number of timers required is greater than '08' (maximum numbers of timers specified in TS 11.14 [4], the card shall return the Status Word '6A80', incorrect parameters in data field, to the Install(Install) command.

6.3.8.1.2 Test suite files

Test Script:	FWK_TIN_TMAL_1.scr
Test Applet:	~~FWK_TIN_TMAL_1.java~~
	~~FWK_TIN_TMAL_2.java~~
	~~FWK_TIN_TMAL_3.java~~
~~Load Script:~~	~~FWK_TIN_TMAL_1.ldr~~
~~Cleanup Script:~~	~~FWK_TIN_TMAL_1.clr~~
~~Parameter File:~~	FWK_TIN_TMAL_1.java
	FWK_TIN_TMAL_2.java
	FWK_TIN_TMAL_3.java
Load Script:	FWK_TIN_TMAL_1.ldr
Cleanup Script:	FWK_TIN_TMAL_1.clr
Parameter File:	FWK_TIN_TMAL_1.par

6.3.8.1.3 Test Procedure

Id	Description	API/Framework Expectation	APDU Expectation

~~1~~	**~~More than 8 timers at the instantiation of applet1: check that applet1 is not installed or that it is not possible to allocate more than 8 timers.~~**
	
	`~~Install for install of applet1 with maximum 9 timers allocated.~~`
	
	`~~applet1 is triggered: we allocate 9 timers~~`
	
	`~~applet1 is selected~~`
		~~Shall throw a ToolkitException with reason NO_TIMER_AVAILABLE only on the 9th allocateTimer()~~	~~The SIM answers to the Envelope with status words 90 00~~
			
			~~2 behaviours may be expected :~~
			~~1. applet1 is not found, status word 6X XX~~
			~~2. applet1 has been installed and only 8 timers are allocated~~

1	**More than 8 timers at the instantiation of applet1: check that applet1 is not installed.** `Install for install of applet1 with maximum 9 timers allocated, requesting a PoR to be sent via SMS-DELIVER-REPORT.`		The SIM answers to the Envelope with status words 9Fxx A GET RESPONSE is sent and the additional data in the PoR is checked. It must be 01 6A 80.
	~~Reset the card and delete instance of applet1~~		
	Reset the card		
2	**Good installation of applet2** `Install for install of applet2 (maximum 4 timers allocated).`		The SIM answers to the Envelope with status words 90 00
3	**Allocate 4 timers Applet2**	No exception shall be thrown.	
4	**Allocate one more timer Applet2**	Shall throw a ToolkitException with reason NO_TIMER_AVAILABLE	
5	**Good installation of applet3** `Install for install of applet3 (maximum 8 timers allocated).`		The SIM answers to the Envelope with status words 90 00
6	**Allocate 4 timers Applet3**	No exception shall be thrown.	
7	**Allocate one more timer Applet3**	Shall throw a ToolkitException with reason NO_TIMER_AVAILABLE	
8	**Check that each timerId (allocated by applet2 and applet3) is between 1 and 8 and is different from each other**		

6.3.8.1.4 Test Coverage

CRR number	Test case number
N1	2, 3, 8
C1	1, 7
C2	4, 5, 6

6.3.8.8 Channel Allocation

Test Area Reference: FWK_TIN_CHAL

6.3.8.7.1 Conformance Requirements

6.3.8.7.1.1 Normal execution

CRRN1: One toolkit applet can register to several channels, but a channel can only be allocated to one toolkit applet.

6.3.8.7.1.2 Context errors

CRRC1 : Allocated channels shall not exceed the maximum number of channels allowed for this applet instance.

CRRC2 : The total number of channels allocated for all the applets shall not exceed 7. If the maximum number of channels required is greater than '07' (maximum numbers of channels specified in TS 11.14 [4]), the card shall return the Status Word '6A80', incorrect parameters in data field, to the Install(Install) command.

6.3.8.7.2 Test suite files

Test Script:	FWK_TIN_CHAL_1.scr
Test Applet:	FWK_TIN_CHAL_1.java
	FWK_TIN_CHAL_2.java
	FWK_TIN_CHAL_3.java
Load Script:	FWK_TIN_CHAL_1.ldr
Cleanup Script:	FWK_TIN_CHAL_1.clr
Parameter File:	FWK_TIN_CHAL_1.par

6.3.8.7.3 Test Procedure

Id	Description	API/Framework Expectation	APDU Expectation

1	**More than 7 channels at the instantiation of applet1: check that applet1 is not installed**
	
	`1-Install for install of applet1 with maximum 8 channels allocated.`
	`A PoR is asked to be sent via SMS-DELIVER-REPORT.`
			1- The SIM answers to the Envelope with status words 9Fxx.
			A GET RESPONSE is sent and the additional data in the PoR is checked. It must be 01 6A 80.

	Reset the card		

2	**Good installation of applet2** `Install for install of applet2 (maximum 4 channels allocated).`		The SIM answers to the Envelope with status words 90 00
3	**Open 4 channels Applet2** `Applet2 builds a proactive command OPEN CHANNEL 4 times, calling init() and send() methods.`	No exception shall be thrown.	OPEN CHANNEL proactive command are fetched. Successful TERMINAL RESPONSE of OPEN CHANNEL are sent to the SIM with Channel Id = 01 to 04
4	**Open one more channel Applet2** `Applet2 builds a proactive command OPEN CHANNEL once again, calling init() and send() methods.`	Shall throw a ToolkitException with reason COMMAND_NOT_ALLOWED	
5	**Good installation of applet3** `Install for install of applet3 (maximum 7 channels allocated).`		The SIM answers to the Envelope with status words 90 00
6	**Open 3 channels Applet3** `Applet3 builds a proactive command OPEN CHANNEL 3 times, calling init() and send() methods.`	No exception shall be thrown.	OPEN CHANNEL proactive command is fetched. Successful TERMINAL RESPONSE of OPEN CHANNEL are sent to the SIM with Channel Id from 05 to 07
7	**Open one more channel Applet3** `Applet3 builds a proactive command OPEN CHANNEL once again, calling init() and send() methods.`	No exception shall be thrown.	OPEN CHANNEL proactive command is fetched. Unsuccessful Terminal Response is sent to the SIM with 'No Channel Available' as Additional Information on Result.

6.3.8.1.4 Test Coverage

CRR number	**Test case number**
N1	2,3
C1	1, 7
C2	4,5,6

6.3.8.8 Minimum Security Level

Test Area Reference: FWK_TIN_MSL

6.3.8.8.1 Conformance Requirements

6.3.8.8.1.1 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 of security 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.

CRRN6: If no Minimum Security Level field is present (no MSL length, no MSL parameter and no MSL data), no minimum security level check shall be performed by the receiving entity.

CRRN7: If the Maximum number of channels field is included in the command data then the Length of Minimum Security Level field shall also be included.

CRRN8: If an optional parameter is included, then all the previous parameters shall be included also

6.3.8.8.2 Test suite files

Test Script:	FWK_TIN_MSL_1.scr
Test Applet:	FWK_TIN_MSL_1.java
Load Script:	FWK_TIN_MSL_1.ldr
Cleanup Script:	FWK_TIN_MSL_1.clr
Parameter File:	FWK_TIN_MSL_1.par

6.3.8.8.3 Test Procedure

Id	Description	API/Framework Expectation	APDU Expectation

1	**Installation with MSL length of 0**
	
	`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- 9000

2	**Installation without MSL field**
	
	`1- Install (install) applet without MSL field (no MSL length, no MSL parameter and no data)`
	`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- 9000

6.3.8.9.4 Test Coverage

CRR number	Test case number
CRRN1	Not applicable
CRRN2	Not applicable
CRRN3	Not applicable
CRRN4	Not applicable
CRRN5	1
CRRN6	2
CRRN7	Not testable
CRRN8	Not testable

6.3.11 Concatenated SMS

6.3.11.1 Concatenation processing

6.3.11.1 Conformance Requirements:

6.3.11.1.1 Normal execution

CRRN1: The SIM Toolkit Framework shall link single Short Messages together to re-assemble the original message before any further processing.

CRRN2: The concatenation control headers used to re-assemble 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 EnvelopeHandler.

CRRN5: The SIM Toolkit Framework shall be able to process messages with the following properties:
The Information Element Identifier is equal to the 8-bit reference number
It contains uncompressed 8 bit data or uncompressed UCS2 data.

6.3.11.2 Test Suite Files

Test Script:	FWK_CSM_PROC_1.scr
Test Applet:	FWK_CSM_PROC_1.java
Load Script:	FWK_CSM_PROC_1.ldr
Cleanup Script:	FWK_CSM_PROC_1.clr
Parameter File:	FWK_CSM_PROC_1.par

6.3.11.3 Test Procedure

Id	Description	API/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`
	
	`A concatenated formatted SMS_PP short message is sent to the SIM (composed of two segments).`		

1	`The second segment of a concatenated short message is sent to the SIM.`	Applet is not triggered.	
2	`The first segment of the concatenated short message is sent to the SIM`	Applet is triggered.	
3	`Call the EnvelopeHanlder.getTheHandler()`	No exception is thrown.	
4	`Call the EnvelopeHandler.findTLV()to select the Dev Id, the adress and the TPDU TLV and the EnvelopeHandler.compareValue() to check each content.`	Check that the message has been re-assembled 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.	

5	A new concatenated formatted short message is sent to the SIM composed of two segments. The Address field of the first segment is different from the address field in the second segment.	Applet is triggered.	
6	Call the EnvelopeHandler.getTheHandler()	No exception is thrown.	
7	Call the EnvelopeHandler.findTLV()to select the the address TLV and the EnvelopeHandler.compareValue() to check its content.	Check that the address field of the message is equal to the address field of the second segment.	
8	A new concatenated formatted short message is sent to the SIM composed of two segments. Some TP_elements of the TP_DU of the first segment are different from the TP elements in the second segment.	Applet is triggered.	
9	Call the EnvelopeHandler.getTheHandler()	No exception is thrown.	
10	Call the EnvelopeHandler.findTLV()to select the the TP DU TLV and the EnvelopeHandler.compareValue() to check its TP elements.	Check that the TP elements of the message are equal to the ones of the second segment.	
11	Send a concatenated formatted short message (composed of 2 segment)with uncompressed 8 bits data.	Applet is triggered.	
	Applet registration to EVENT_UNFORMATTED_SMS_PP_ENV and triggering Same test as 1 but with an unformatted SMS_PP envelope. A concatenated unformatted SMS_PP short message is sent to the SIM (composed of two segments).		
12	The second segment of a concatenated short message is sent to the SIM.	Applet is not triggered.	
13	The first segment of the concatenated short message is sent to the SIM	Applet is triggered.	
14	Call the EnvelopeHanlder.getTheHandler()	No exception is thrown.	
15	Call the EnvelopeHandler.findTLV()to select the Dev Id, the adress and the TPDU TLV and the EnvelopeHandler.compareValue() to check each content.	Check that the message has been re-assembled 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.	
16	A new concatenated formatted short message is sent to the SIM composed of two segments. The Address field of the first segment is different from the address field in the second segment.	Applet is triggered.	
17	Call the EnvelopeHandler.getTheHandler()	No exception is thrown.	
18	Call the EnvelopeHandler.findTLV()to select the the address TLV and the EnvelopeHandler.compareValue() to check its content.	Check that the address field of the message is equal to the address field of the second segment.	
19	A new concatenated unformatted short message is sent to the SIM composed of two segments. Some TP_elements of the TP_DU of the first segment are different from the TP_elements in the second segment.	Applet is triggered.	
20	Call the EnvelopeHandler.getTheHandler()	No exception is thrown.	
21	Call the EnvelopeHandler.findTLV()to select the the TP DU TLV and the EnvelopeHandler.compareValue() to check its TP elements.	Check that the TP elements of the message are equal to the ones of the second segment.	
22	Send a concatenated unformatted short message (composed of 2 segments) with uncompressed UCS2 data.	Applet is triggered.	

6.3.11.4 Test Coverage

CRR number	Test case number
N1	1,2, 3, 5, 6, 8, 9, 12, 13, 14, 16, 17, 19, 20
N2	4,15
N3	7,10, 18, 21
N4	4,15
N5	11,22

Annex A (normative): Class and Methods AID numbering and acronyms

A.2.4 EnvelopeHandler methods

Method Name	Acronyms	Numbering on 6 bits
~~byte~~ Byte getEnvelopeTag()	GENT	000001
~~byte~~ Byte getItemIdentifier()	GIID	000010
~~short~~ Short getSecuredDataLength()	GSDL	000011
~~short~~ Short getSecuredDataOffset()	GSDO	000100
EnvelopeHandler getTheHandler()	GTHD	000101
~~short~~ Short getTPUDLOffset()	GTPO	000110
Short getCapacity()	GCAP	010010
Short getUserDataLength()	GUDL	010011
Byte getChannelIdentifier()	GCID	010100
Inherited Method Name: ViewHandler		
Byte compareValue(short valueOffset,byte[] compareBuffer, short compareOffset, short compareLength)	CPRVS_BSS	000111
Short copy(byte[] dstBuffer,short dstOffset,short dstLength)	COPY_BSS	001000
Short copyValue(short valueOffset, byte[] dstBuffer,short dstOffset,short dstLength)	CPYVS_BSS	001001
Byte findAndCompareValue(byte tag,byte[] compareBuffer,short compareOffset)	FACRB_BS	001010
Byte findAndCompareValue(byte tag,byte occurrence, short valueOffset,byte[] compareBuffer,short compareOffset,short compareLength)	FACRBBS_BSS	001011
Short FindAndCopyValue(byte tag,byte occurrence,short valueOffset, byte[] dstBuffer, short dstOffset, short dstLength)	FACYBBS_BSS	001100
Short findAndCopyValue(byte tag,byte[] dstBuffer,short dstOffset)	FACYB_BS	001101
Byte FindTLV(byte tag,byte occurrence)	FINDBB	001110
Short GetLength()	GLEN	001111
Byte GetValueByte(short valueOffset)	GVBYS	010000
Short GetValueLength()	GVLE	010001

A.2.5 EnvelopeResponseHandler methods

Method Name	Acronym	Numbering on 6 bits
`EnvelopeResponseHandler getTheHandler()`	GTHD	000001
`Void post(byte statusType)`	POSTB	000010
`Void postAsBERTLV(byte statusType, byte tag)`	POSTBB	000011
`Short getCapacity()`	GCAP	010101
Inherited Method Name: EditHandler		
`Void appendArray(byte[] buffer, short offset, short length)`	APDA_BSS	000100
`Void appendTLV(byte tag, byte value)`	APTLBB	000101
`Void appendTLV(byte tag, byte[] value, short valueOffset, short valueLength)`	APTLB_BSS	000110
`Void appendTLV(byte tag, byte value1, byte value2)`	APTLBBB	000111
`Void appendTLV(byte tag, byte value1, byte[] value2, short value2Offset, short value2Length)`	APTLBB_BSS	001000
`Void clear()`	CLER	001001
Inherited Method Name: ViewHandler		
`Byte compareValue(short valueOffset,byte[] compareBuffer, short compareOffset, short compareLength)`	CPRVS_BSS	001010
`Short Copy(byte[] dstBuffer,short dstOffset,short dstLength)`	COPY_BSS	001011
`Short CopyValue(short valueOffset, byte[] dstBuffer,short dstOffset,short dstLength)`	CPYVS_BSS	001100
~~`Byte FindAndCompareValue(byte tag,byte[] compareBuffer,short compareOffset)`~~	~~FACRB_BS~~	~~001101~~
`Byte findAndCompareValue(byte tag,byte[] compareBuffer,short compareOffset)`	FACRB_BS	001101
`Byte findAndCompareValue(byte tag,byte occurence, short valueOffset,byte[] compareBuffer,short compareOffset,short compareLength)`	FACRBBS_BSS	001110
~~`Short FindAndCopyValue(byte tag,byte occurence,short valueOffset, byte[] dstBuffer, short dstOffset, short dstLength)`~~	~~FACYBBS_BSS~~	~~001111~~
`Short findAndCopyValue(byte tag,byte occurence,short valueOffset, byte[] dstBuffer, short dstOffset, short dstLength)`	FACYBBS_BSS	001111
`Short findAndCopyValue(byte tag,byte[] dstBuffer,short dstOffset)`	FACYB_BS	010000
~~`Byte FindTLV(byte tag,byte occurrence)`~~	~~FINDBB~~	~~010001~~
`Byte findTLV(byte tag,byte occurrence)`	FINDBB	010001
`Short GetLength()`	GLEN	010010
~~`Byte GetValueByte(short valueOffset)`~~	~~GVBYS~~	~~010011~~
`Byte getValueByte(short valueOffset)`	GVBYS	010011
~~`Short GetValueLength()`~~	~~GVLE~~	~~010100~~
`Short getValueLength()`	GVLE	010100

A.2.7 ProactiveHandler methods

Method Name	Acronyms	Numbering on 6 bits
~~GetTheHandler()~~	~~GTHD~~	~~000001~~
ProactiveHandler getTheHandler()	GTHD	000001
~~Init(byte type, byte qualifier, byte dstDevice)~~	~~INITBBB~~	~~000010~~
Void init(byte type, byte qualifier, byte dstDevice)	INITBBB	000010
~~InitDisplayText(byte qualifier, byte dcs, byte[] buffer, short offset, short length)~~	~~INDTBB_BSS~~	~~000011~~
Void initDisplayText(byte qualifier, byte dcs, byte[] buffer, short offset, short length)	INDTBB_BSS	000011
~~InitGetInkey(byte qualifier, byte dcs, byte[] buffer, short offset, short length)~~	~~INGKBB_BSS~~	~~000100~~
Void initGetInkey(byte qualifier, byte dcs, byte[] buffer, short offset, short length)	INGKBB_BSS	000100
~~InitGetInput(byte qualifier, byte dcs, byte[] buffer, short offset, short length, short minRespLength, short maxRespLength)~~	~~INGPBB_BSSSS~~	~~000101~~
Void initGetInput(byte qualifier, byte dcs, byte[] buffer, short offset, short length, short minRespLength, short maxRespLength)	INGPBB_BSSSS	000101
Byte send()	SEND	000110
Short getCapacity()	GCAP	011000
Void initCloseChannel(byte bChannelIdentifier)	ICCHB	011001
Inherited Method Name: EditHandler		
Void appendArray(byte[] buffer, short offset, short length)	APDA_BSS	000111
Void appendTLV(byte tag, byte value)	APTLBB	001000
Void appendTLV(byte tag, byte[] value, short valueOffset, short valueLength)	APTLB_BSS	001001
Void appendTLV(byte tag, byte value1, byte value2)	APTLBBB	001010
Void appendTLV(byte tag, byte value1, byte[] value2, short value2Offset, short value2Length)	APTLBB_BSS	001011
Void clear()	CLER	001100

Inherited Method Name: ViewHandler		
~~Byte CompareValue(short valueOffset,byte[] compareBuffer,short compareOffset, short compareLength)~~	~~CPRVS_BSS~~	~~001101~~
Byte compareValue(short valueOffset,byte[] compareBuffer, short compareOffset, short compareLength)	CPRVS_BSS	001101
~~Short Copy(byte[] dstBuffer,short dstOffset,short dstLength)~~	~~COPY_BSS~~	~~001110~~
Short copy(byte[] dstBuffer,short dstOffset,short dstLength)	COPY_BSS	001110
~~Short CopyValue(short valueOffset, byte[] dstBuffer,short dstOffset,short dstLength)~~	~~CPYVS_BSS~~	~~001111~~
Short copyValue(short valueOffset, byte[] dstBuffer,short dstOffset,short dstLength)	CPYVS_BSS	001111
~~Byte FindAndCompareValue(byte tag,byte[] compareBuffer,short compareOffset)~~	~~FACRB_BS~~	~~010000~~
Byte findAndCompareValue(byte tag,byte[] compareBuffer,short compareOffset)	FACRB_BS	010000
~~-~~Byte findAndCompareValue(byte tag,byte occurence, ~~–~~ short valueOffset,byte[] compareBuffer,short compareOffset,short compareLength)	FACRBBS_BSS	010001
~~Short FindAndCopyValue(byte tag,byte occurence,short valueOffset, byte[] dstBuffer, short dstOffset, short dstLength)~~	~~FACYBBS_BSS~~	~~010010~~
Short findAndCopyValue(byte tag,byte occurence,short valueOffset, byte[] dstBuffer, short dstOffset, short dstLength)	FACYBBS_BSS	010010
~~-~~Short findAndCopyValue(byte tag,byte[] dstBuffer,short dstOffset)	FACYB_BS	010011
~~Byte FindTLV(byte tag,byte occurrence)~~	~~FINDBB~~	~~010100~~
Byte findTLV(byte tag,byte occurrence)	FINDBB	010100
~~Short GetLength()~~	~~GLEN~~	~~010101~~
Short getLength()	GLEN	010101
~~Byte GetValueByte(short valueOffset)~~	~~GVBYS~~	~~010110~~
Byte getValueByte(short valueOffset)	GVBYS	010110
~~Short GetValueLength()~~	~~GVLE~~	~~010111~~
Short getValueLength()	GVLE	010111

A.2.8 ProactiveResponseHandler methods

Method Name	Acronyms	Numbering on 6 bits
~~Short CopyAdditionalInformation(byte[] dstBuffer, short dstOffset, short dstLength)~~	~~CPAI_BSS~~	~~000001~~
Short copyAdditionalInformation(byte[] dstBuffer, short dstOffset, short dstLength)	CPAI_BSS	000001
Short copyTextString(byte[] dstBuffer, short dstOffset)	CPTS_BS	000010
Short getAdditionalInformationLength()	GTIL	000011
Byte getGeneralResult()	GTGR	000100
Byte getItemIdentifier()	GTII	000101
Byte getTextStringCodingScheme()	GTCS	000110
Short getTextStringLength()	GTTL	000111
~~GetTheHandler()~~	~~GTHD~~	~~001000~~
ProactiveResponseHandler getTheHandler()	GTHD	001000
Short getCapacity()	GCAP	010100
Byte getChannelIdentifier()	GCID	010101
Short copyChannelData(byte[] dstBuffer, short dstOffset, short dstLength)	CCHD_BSS	010110

Inherited Method Name: ViewHandler		
Byte CompareValue(short valueOffset,byte[] compareBuffer,short compareOffset, short compareLength)	CPRVS_BSS	001001
Short Copy(byte[] dstBuffer,short dstOffset,short dstLength)	COPY_BSS	001010
Short CopyValue(short valueOffset, byte[] dstBuffer,short dstOffset,short dstLength)	CPYVS_BSS	001011
Byte FindAndCompareValue(byte tag,byte[] compareBuffer,short compareOffset)	FACRB_BS	001100
Byte findAndCompareValue(byte tag,byte occurence, short valueOffset,byte[] compareBuffer,short compareOffset,short compareLength)	FACRBBS_BSS	001101
Short FindAndCopyValue(byte tag,byte occurence,short valueOffset, byte[] dstBuffer, short dstOffset, short dstLength)	FACYBBS_BSS	001110
Short findAndCopyValue(byte tag,byte[] dstBuffer,short dstOffset)	FACYB_BS	001111
Byte FindTLV(byte tag,byte occurrence)	FINDBB	010000
Short GetLength()	GLEN	010001
Byte GetValueByte(short valueOffset)	GVBYS	010010
Short GetValueLength()	GVLE	010011

B.4 Style and formatting

In order to~~ ~~ show a common appearance all the scripts shall follow those format rules:

- start always with a 'RST'~~ followed by an 'INI' command~~.
- The command, data to be checked and status to be checked shall be presented in the following order:

  CMD *COMMAND* [*EXPECTED DATA*] (*EXPECTED STATUS*)

– APDU shall be presented with command (CLA INS P1 P2 P3) in one line and data (if present) in next line grouped ~~-~~16 bytes per line (see example above).

– The expected data (if present) shall be presented in 16 bytes groups per line (see example above).

F.1 Toolkit Installation Parameters (TIN)

Test Area within the chapter	Acronyms	Numbering on 6 bits
Timer allocation	TMAL	000001
Item identifier	ITID	000010
Item position	ITPO	000011
Access conditions	ACCO	000100
Priority level	PRLV	000101
Maximum length for each menu entry	MLME	000110
Number of menu entries	NBME	000111
Memory space	MESP	001000
Channel Allocation	CHAL	001001
Minimum Security Level	MSL	001010

F.3 Handler Integrity (HIN)

Test Area within the chapter	Acronyms	Numbering on 6 bits
ProactiveHandler	PAHD	000001
ProactiveResponseHandler	PRHD	000010
EnvelopeHandler	ENHD	000011
~~RFU (~~EnvelopeResponseHandler~~)~~	~~(~~ERHD~~)~~	000100

F.4 Applet Triggering (APT)

Test Area within the chapter	Acronyms	Numbering on 6 bits
EVENT_PROFILE_DOWNLOAD	EPDW	000001
EVENT_MENU_SELECTION	EMSE	000010
EVENT_MENU_SELECTION_HELP_REQUEST	EMSH	000011
EVENT_FORMATTED_SMS_PP_ENV	EFSE	000100
EVENT_UNFORMATTED_SMS_PP_ENV	EUSE	000101
EVENT_CALL_CONTROL_BY_SIM	ECCN	000110
EVENT_MO_SHORT_MESSAGE_CONTROL_BY_SIM	EMCN	000111
EVENT_TIMER_EXPIRATION	ETEX	001000
EVENT_UNFORMATTED_SMS_CB	EUCB	001001
EVENT_EVENT_DOWNLOAD_MT_CALL	EDMC	001010
EVENT_EVENT_DOWNLOAD_CALL_CONNECTED	EDCC	001011
EVENT_EVENT_DOWNLOAD_CALL_DISCONNECTED	EDCD	001100
EVENT_EVENT_DOWNLOAD_LOCATION_STATUS	EDLS	001101
EVENT_EVENT_DOWNLOAD_USER_ACTIVITY	EDUA	001110
EVENT_EVENT_DOWNLOAD_IDLE_SCREEN_AVAILABLE	EDIS	001111
EVENT_EVENT_DOWNLOAD_CARD_READER_STATUS	EDCR	010000
~~RFU (EVENT_UNRECOGNIZED_ENVELOPE)~~	~~(EUEN)~~	~~010001~~
EVENT_UNRECOGNIZED_ENVELOPE	EUEV	010001
EVENT_STATUS_COMMAND	ESTC	010010
EVENT_EVENT_DOWNLOAD_LANGUAGE_SELECTION	EDLG	010011
EVENT_EVENT_DOWNLOAD_BROWSER_TERMINATION	EDBT	010100
EVENT_FORMATTED_SMS_CB	EFCB	010101
EVENT_FIRST_COMMAND_AFTER_SELECT	EFCA	010110
EVENT_EVENT_DOWNLOAD_DATA_AVAILABLE	EDDA	010111
EVENT_EVENT_DOWNLOAD_CHANNEL_STATUS	EDCS	011000
EVENT_FORMATTED_SMS_PP_UPD	EFSU	011001
EVENT_UNFORMATTED_SMS_PP_UPD	EUSU	011010

F.5 Proactive Command Sending (PCS)

Test Area within the chapter	Acronyms	Numbering on 6 bits
System Proactive commands	SPCO	000001
Interaction with GSM commands	IGCO	000010
Errors during proactive command sending	EPCS	000011
Proactive Command Control	PCCO	000100

F.6 Envelope Response Posting (ERP)

Test Area within the chapter	Acronyms	Numbering on 6 bits
EVENT_CALL_CONTROL_BY_SIM	ECCN	000001
EVENT_MO_SHORT_MESSAGE_CONTROL_BY_SIM	EMCN	000010
EVENT_UNRECOGNIZED_ENVELOPE	EUEN	000011
EVENT_FORMATTED_SMS_PP_ENV	EFSE	000010

F.11 Concatenation processing (PROC)

Test Area within the chapter	Acronyms	Numbering on 6 bits
Concatenation processing	PROC	000001

G.2.5 INSTALL(install) Section

Here are the parameters to be included in the Install(Install) command (as specified in ~~GSM 03.48~~ TS 23.048 [8])

Parameter	Description
PackageAID	AID of the package
AppletClassAID	AID of the applet
InstanceAID	AID of the instance of the applet
InstallationNonVolatileMemSize	Non volatile memory required for installation, in bytes
InstallationVolatileMemSize	Volatile memory required for installation, in bytes
AccessDomain	Specify the SIM files that may be accessed by the applet and the operations allowed on these files. This parameter includes the Access Domain Parameter (ADP) and Access Domain Data (ADD)
PriorityLevel	Priority level of the Toolkit applet instance
MaxNumberOfTimers	Maximum number of timers allowed for this applet instance
MaxMenuEntryTextLength	Maximum text length for a menu entry
MaxNumberOfMenuEntries	Maximum number of menu entries allowed for this applet instance
MenuEntriesPositionIdentifier	For each menu entry: Position and identifier of that menu entry
MaxNumberOfChannels	Maximum Number of channels for this applet instance
MSLFieldLength	Length of Minimum Security Level field
MSLParameter	MSL Parameter
MSLData	MSL Data
AppletSpecificParameters	Parameters specific to the applet

The applet shall be installed with install(install and make selectable) command.

G.3 Full example

```
[CONVERT]

PackageAID = A0 00 00 00 30 00 02 FF FF FF FF 89 00 00 01 00

PackageName = sim.test.access.api_1_svw_updrbs

PackageVersion = 1.0
```

```
AppletClassAID = A0 00 00 00 30 00 02 FF FF FF FF 89 00 00 01 01
AppletClassName = API_1_SVW_UPDRBS_1
AppletClassAID = A0 00 00 00 30 00 02 FF FF FF FF 89 00 00 01 02
AppletClassName = API_1_SVW_UPDRBS_2

[INSTALL(load)]
PackageNonVolatileMemSize = 0D27
;InstallationNonVolatileMemSize = 0400
;InstallationVolatileMemSize = 0000

[LOAD]
MaxLoadCommandDataLength = 6C ; max value

[INSTALL(install)]
AppletClassAID = A0 00 00 00 30 00 02 FF FF FF FF 89 00 00 01 01
InstanceAID =    A0 00 00 00 30 00 02 FF FF FF FF 89 00 00 01 01
InstallationNonVolatileMemSize = 0400
InstallationVolatileMemSize = 0000
AccessDomain = 00
PriorityLevel = FF
MaxNumberOfTimers = 00
MaxMenuEntryTextLength = 10
MaxNumberOfMenuEntries = 01
MenuEntriesPositionIdentifier = 0001
AppletSpecificParameters =

[INSTALL(install)]
AppletClassAID = A0 00 00 00 30 00 02 FF FF FF FF 89 00 00 01 02
InstanceAID =    A0 00 00 00 30 00 02 FF FF FF FF 89 00 00 01 02
InstallationNonVolatileMemSize = 0200
InstallationVolatileMemSize = 0000
MenuEntriesPositionIdentifier = 0002
MaxNumberOfChannels = 05
MSLFieldLength = 00
MSLParameter =
MSLData =
```

```
; rest of INSTALL(install) parameters are taken from previous INSTALL(install)...
```