Open Mobile API Test Specification for Transport API

V1.1

Table of Contents

Table of Tables

1. Terminology

The given terminology is used in this document.

1.1 Abbreviations and notations

Abbreviation	Description
SE	Secure Element
API	Application Programming Interface
ATR	Answer to Reset (as per ISO/IEC 7816-4)
APDU	Application Protocol Data Unit (as per ISO/IEC 7816-4)
ISO	International Organization for Standardization
ASSD	Advanced Security SD cards (SD memory cards with an embedded security system) as specified by the SD Association.
OS	Operating System
RIL	Radio Interface Layer
SFI	Short File ID
FID	File ID
FCP	File Control Parameters
MF	Master File
DF	Dedicated File
EF	Elementary File
OID	Object Identifier
PPS	Protocol Parameter Selection (as per ISO/IEC 7816-4)
DER	Distinguished Encoding Rules of ASN.1
ASN.1	Abstract Syntax Notation One
DUT	Device Under Test
CMD	The APDU command sent from the DUT
RESP	The APDU response sent to the DUT

Table 1: Abbreviations and Notations

1.2 Terms

Term	Description
Secure Element	A Secure Element (SE) is a tamper-resistant component which is used to provide the security, confidentiality, and multiple application environments required to support various business models. For example UICC/SIM, embedded Secure Element, Secure SD card.
Applet	General term for Secure Element application: An application which is installed in the SE and runs within the SE. For example a JavaCard™ application or a native application.
Application	Device/terminal/mobile application: An application which is installed in the mobile device and runs within the mobile device

Session	An open connection between an application on the device (e.g. mobile phone) and a SE.
Channel	An open connection between an application on the device (e.g. mobile phone) and an applet on the SE.
restarted	The DUT has been switched off completely and has been started again. No quick start, soft power off, or similar.
same object	Two objects are the same object if the language-specific mechanism to check for identity of objects indicates that they are the same object. For example, for Java, the == operator should be used.
No APDU	When "No APDU" is mentioned in a test case, it means that the device shall not send any command to the Secure Element while processing the called API method. For a test tool, it means that no APDU (except STATUS command generated by the device telecom stack in case of UICC) shall be sent on the device-SE interface on any channel.
none	When "none" is mentioned in a test case, it means that it is not relevant if APDUs are sent.
No selection	No select by DF name command is sent.

Table 2: Terms

1.3 Format of the table of optional features and applicability table

The columns in tables 4 and 5 for the optional features have the following meaning:

Column	Meaning
Option	The optional feature supported or not by the DUT
Status	• OP - optional feature
Optional item	The mnemonic identifiers for each optional item

The columns in the applicability table 6 have the following meaning:

Column	Meaning
Clause	Reference to the clause index in the document
Test case number and description	The test case description in the document
SUE	The support of the tested feature/method for the Simulated Environment has the following status: • M mandatory - the capability is required to be supported. • OP optional - the capability may be supported or not. In case the support is declared by terminal, the test shall be executed. • N/A not applicable - in the given context, it is impossible to use the capability.

RSE

The support of the tested feature/method for the Real SE Environment has the following status:

- M mandatory - the capability is required to be supported.
- OP optional - the capability may be supported or not. In case the support is declared by terminal, the test shall be executed.
- N/A not applicable - in the given context, it is impossible to use the capability.

2. Informative References

Specification	Description
[1] OMAPI v2.05	SIMalliance Open Mobile API specification v2.05
[2] GP 2.2	GlobalPlatform Card Specification v2.2
[3] ISO/IEC 7816-4:2005	Identification cards - Integrated circuit cards - Part 4: Organisation, security and commands for interchange
[4] ISO/IEC 7816-5:2004	Identification cards - Integrated circuit cards - Part 5: Registration of application providers
[5] ISO/IEC 7816-15:2004	Identification cards - Integrated circuit cards with contacts - Part 15: Cryptographic information application
[6] PKCS #11 v2.20	Cryptographic Token Interface Standard Go to following website for PKCS#15 documentation: http://www.rsa.com/rsalabs/node.asp?id=2133
[7] PKCS #15 v1.1	Cryptographic Token Information Syntax Standard
[8] Java™ Cryptography Architecture API Specification & Reference	Go to the following website for JCA documentation: http://download.oracle.com/javase/1.4.2/docs/guide/security/CryptoSpec.html
[9] ISO/IEC 8825-1:2002 | ITU-T Recommendations X.690 (2002)	Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)
[10] GlobalPlatform Secure Element Access Control, v1.0	Specification for controlling access to Secure Elements based on access policies that are stored in the Secure Element

Table 3: Informative References

3. Overview

This test specification describes how to test the Transport API part of the Open Mobile API. This is the mandatory part of the Open Mobile API. The other parts of the Open Mobile API shall be tested in a similar way.

This test specification is based on v2.05 of the Open Mobile API specification.

4. Applicability

4.1 Applicability of the tests

The test cases are categorised in the applicability table to use the test environment as follow:

- **Simulated UICC environment (SUE):** Test method shall be implemented in a simulated environment for the UICC.
- **Real SE environment (RSE):** Test method shall use a real SE environment. The test method shall use one type of SE, which is determined by implementation in the DUT and the applicability is stated in the table as:
    - UICC: test cases executed with real UICC.
    - eSE: test cases executed with eSE.
    - mSD: test cases executed with mSD.

If both test methods are marked as applicable (SUE and RSE), only one test method is required for demonstrating device compliance. The test cases for Reader, Session and Channel should be executed for each reader supported by the DUT using the environment as defined above.

4.2 Table of DUT options

The DUT supplier shall specify the following information:

- Supported Readers (number and type)

The DUT supplier shall state the support of possible options in table 4 for each SE.

Item	Option	Status	Optional item
1	DUT offers possibility to log APDU communication to eSE or µSD	OP	OP-001
2	access to the basic channel is blocked by the DUT	OP	OP-002
3	access to the basic channel is allowed by the DUT	OP	OP-003
4	the ATR returned by the SE is available	OP	OP-004
5	the ATR returned by the SE is not available	OP	OP-005
6	DUT supports T=0 communication with UICC	OP	OP-006
7	DUT supports T=1 communication with UICC	OP	OP-007
8	The selection response can be retrieved by the reader implementation	OP	OP-008
9	The selection response cannot be retrieved by the reader implementation	OP	OP-009
10	Access to the default applet is allowed by the DUT	OP	OP-010
11	Access to the default applet is blocked by the DUT	OP	OP-011
12	DUT knows when all SE logical channels are already opened	OP	OP-012
13	DUT relies on SE to know if all logical channels are already opened and check access control rules on openSession	OP	OP-013
14	DUT relies on SE to know if all logical channels are already opened and check access control rules on openLogicalChannel	OP	OP-014

Table 4: DUT Options

4.3 Table of test tool options

The used test tool must support the test of all mandatory test cases as well as all DUT options. But there are a few test cases that depend on the support of the test tool. These test tool options are listed below.

The test tool supplier shall state the support of possible options in table 5 for each SE.

Item	Option	Status	Optional item
1	Test tool provides a possibility to interrupt the communication with the SE (this needs to be supported by the device as well)	OP	TTOP-01
2	UICC Simulator supports T=1	OP	TTOP-02
3	The test applet is the default SE Applet	OP	TTOP-03
4	The default applet is different from the test applet, e.g. USIM on a UICC SE, etc.	OP	TTOP-04

Table 5: Test Tool Options

4.4 Applicability table

The following table specifies the applicability of each test case to the mobile.

Clause	Test case number and description	SUE	RSE		
			UICC	eSE	mSD
	class SEService				
6.1.1	Constructor: SEService(Context context, SEService.CallBack listener)	M	M	M	M
6.1.2	Method: Reader[] getReaders()	M	M	M	M
6.1.3	Method: boolean isConnected ()	M	M	M	M
6.1.4	Method: void shutdown () ID1	M	M	M	M
6.1.4	Method: void shutdown () ID2, ID3	M	M	OP-001	OP-001
6.1.5	Method: String getVersion()	M	M	M	M
6.2.1	Method: void serviceConnected(SEService service)	M	M	M	M
	class Reader				
6.3.1	Method: String getName()	M	M	M	M
6.3.2	Method SEService getSEService()	M	M	M	M
6.3.3	Method: boolean isSecureElementPresent() ID1	M	M	M	M
6.3.3	Method: boolean isSecureElementPresent() ID2	M	TTOP-01	TTOP-01	TTOP-01
6.3.4	Method: Session openSession()	M	M	M	M
6.3.5	Method: void closeSessions() ID1	M	M	M	M
6.3.5	Method: void closeSessions() ID2	M	M	OP-001	OP-001
	class Session				
6.4.1	Method: Reader getReader()	M	M	M	M
6.4.2	Method: byte[] getATR() ID1	OP-004	OP-004	OP-004	OP-004
6.4.2	Method: byte[] getATR() ID2	OP-004	OP-004	OP-004 and OP-001	OP-004 and OP-001
6.4.2	Method: byte[] getATR() ID3	OP-005	OP-005	OP-005	OP-005
6.4.3	Method: void close()	M	M	OP-001	OP-001
6.4.4	Method: boolean isClosed()	M	M	M	M
6.4.5	Method: void closeChannels() ID1	M	M	OP-001	OP-001
6.4.5	Method: void closeChannels() ID2	M	M	M	M
6.4.6	Method: Channel openBasicChannel ID1 – ID3, ID5, ID6, ID8, ID9, ID11 – ID13	OP-003	OP-003	M	M
6.4.6	Method: Channel openBasicChannel ID4a	OP-003 and TTOP-03	OP-003 and TTOP-03	TTOP-03	TTOP-03
6.4.6	Method: Channel openBasicChannel ID4b	OP-003 and TTOP-04	OP-003 and TTOP-04	TTOP-04	TTOP-04
6.4.6	Method: Channel openBasicChannel ID7	OP-002	OP-002	N/A	NA
6.4.6	Method: Channel openBasicChannel ID10	OP-003	OP-003 and TTOP-01	TTOP-01	TTOP-01

Clause	Test case number and description	SUE	RSE		
			UICC	eSE	mSD
6.4.7	Method: Channel openLogicalChannel ID1, ID2, ID6, ID7, ID09 – ID17	M	M	M	M
6.4.7	Method: Channel openLogicalChannel ID3a	OP-010	OP-010	M	M
6.4.7	Method: Channel openLogicalChannel ID3b	OP-011	OP-011	N/A	N/A
6.4.7	Method: Channel openLogicalChannel ID4a	OP-010 and TTOP-03	OP-010 and TTOP-03	TTOP-03	TTOP-03
6.4.7	Method: Channel openLogicalChannel ID4b	OP-010 and TTOP-04	OP-010 and TTOP-04	TTOP-04	TTOP-04
6.4.7	Method: Channel openLogicalChannel ID5a	OP-012	OP-012	OP-012	OP-012
6.4.7	Method: Channel openLogicalChannel ID5b	OP-013	OP-013	OP-013	OP-013
6.4.7	Method: Channel openLogicalChannel ID5c	OP-014	OP-014	OP-014	OP-014
6.4.7	Method: Channel openLogicalChannel ID8	M	TTOP-01	TTOP-01	TTOP-01
6.4.8	Method: Channel openLogicalChannel – Extended logical channels ID1	OP-012	OP-012	OP-012	OP-012
6.4.8	Method: Channel openLogicalChannel – Extended logical channels ID2	OP-013	OP-013	OP-013	OP-013
6.4.8	Method: Channel openLogicalChannel – Extended logical channels ID3	OP-014	OP-014	OP-014	OP-014
	class Channel				
6.5.1	Method: void close() ID2	OP-003	OP-003	OP-003	OP-003
6.5.1	Method: void close() ID1,ID3, ID4	M	M	OP-001	OP-001
6.5.2	Method: boolean isBasicChannel() ID1	OP-003	OP-003	M	M
6.5.2	Method: boolean isBasicChannel() ID2	M	M	M	M
6.5.3	Method: boolean isClosed() ID1	M	M	M	M
6.5.3	Method: boolean isClosed() ID2	M	M	OP-001	OP-001
6.5.4	Method: byte[] getSelectResponse() ID1,2,4,5,7,8	OP-008	OP-008	OP-008	OP-008
6.5.4	Method: byte[] getSelectResponse() ID 3	OP-008 and OP-010	OP-008 and OP-010	OP-008	OP-008
6.5.4	Method: byte[] getSelectResponse() ID6	OP-009	OP-009	OP-009	OP-009
6.5.5	Method: Session getSession()	M	M	M	M
6.5.6	Method: byte[] transmit(byte[] command) ID1	OP-003	OP-003	M	M
6.5.6	Method: byte[] transmit(byte[] command) ID2 – ID7; ID9 – ID11,ID15 - ID20, ID23	M	M	M	M
6.5.6	Method: byte[] transmit(byte[] command) ID8,	M	TTOP-01	TTOP-01	TTOP-01
6.5.6	Method: byte[] transmit(byte[] command) ID12	M	NA	NA	NA
6.5.6	Method: byte[] transmit(byte[] command) ID13	OP-006	OP-006	NA	NA
6.5.6	Method: byte[] transmit(byte[] command) ID14	OP-007 and TTOP-02	OP-007	NA	NA
6.5.6	Method: byte[] transmit(byte[] command) ID21, ID22	OP-006	OP-006	M	M
6.5.7	Method: Boolean[] selectNext() ID1 –ID2, ID7	M	M	M	M
6.5.7	Method: Boolean[] selectNext() ID3-ID4, ID8-ID9	OP-008	OP-008	OP-008	OP-008
6.5.7	Method: Boolean[] selectNext() ID5	M	TTOP-01	TTOP-01	TTOP-01
6.5.7	Method: Boolean[] selectNext() ID6	M	NA	NA	NA

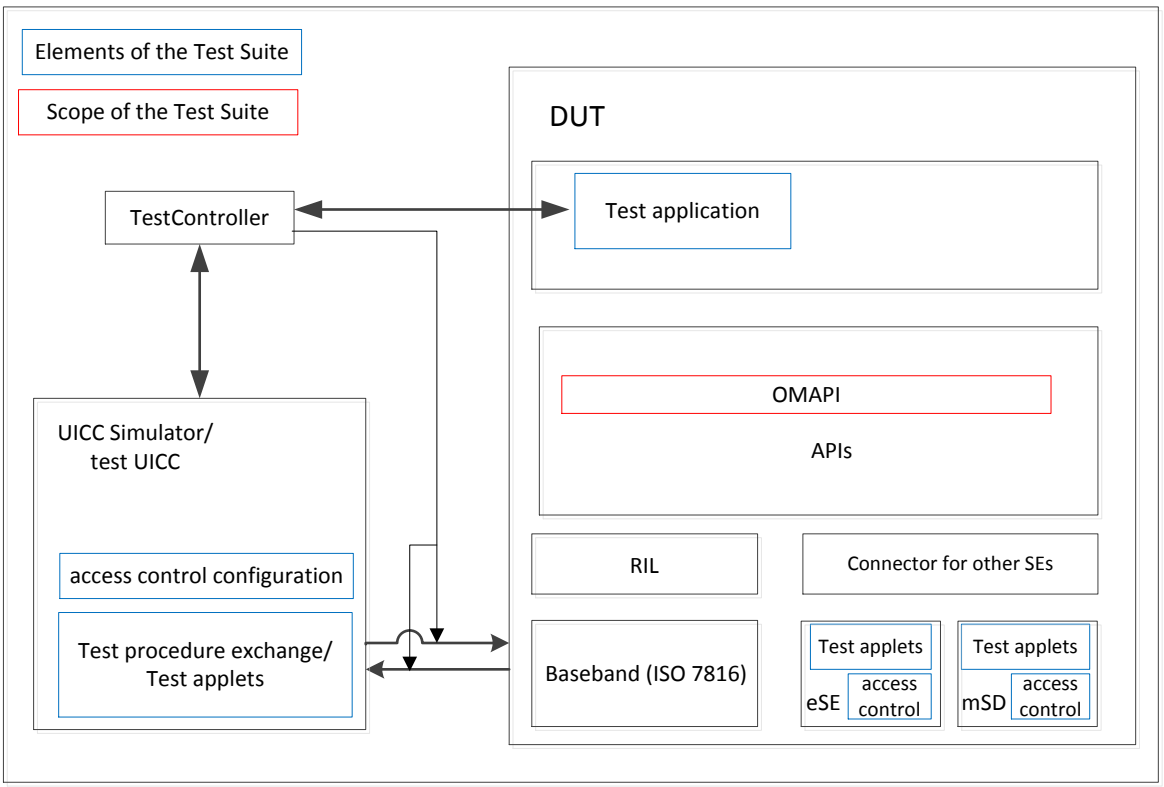
Table 6: Applicability of Tests

5. Test Environment

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

5.1 Test environment description

The general architecture for the test environment is:



5.2 Test tool

SIMalliance Open Mobile Test Specification v1.1 integrates test cases targeting classical APDU commands (excluding extended APDU commands) so the test tools do not need to manage extended APDU commands for this release. Please note that this will change in the next test specification release as SIMalliance is planning to integrate new test cases targeting this feature.

5.2.1 UICC Simulator

The test equipment used for executing this test specification shall meet the following requirements in order to be able to use the OMAPI implementation on a mobile device:

- be able to send and receive the commands correctly on the lower layer; i.e. to use commands as specified in ISO/IEC 7816-4.
- be able to provide access to Basic and Logical Channels for APDUs transmission and channels can be opened simultaneously.

- the ATR used by the test equipment shall correctly show the minimum capability required to run the tests.
- shall be capable to work in a multi SE environment.
- shall be able to provide the access control conditions according to section 5.8.
- for the implementation of the test procedure exchange/test Applets and the access control configuration, the main source of reference is this test specification.
- shall support 20 channels (including the basic channel).

5.2.2 UICC, eSE and mSD

Unless otherwise specified, the following requirements and configuration shall be met:

- be able to send and receive the commands correctly on the lower layer; i.e. to use commands as specified in ISO/IEC 7816-4.
- be able to provide access to Basic and Logical Channels for APDUs transmission and channels can be opened simultaneously.
- the ATR sent by the SE shall correctly show the minimum capability required to run the tests.
- shall be capable to work in a multi SE environment.
- shall be able to provide the access control conditions according to section 5.8.
- all the test applets specified in section 5.7 need to be installed on the SE.
- only one NAA (network authentication application) installed to avoid that the mobile opens logical channels.
- it shall be possible to verify APDU communication in a reliable way.
- shall support 20 channels (including the basic channel).

5.2.3 Test controller

The following requirements shall be provided by the test controller:

- the APDU exchange must be made visible by the test tool when they are available. For example in the case of UICC, or UICC simulator.
- the API commands must be made visible by the test tool.
- shall provide the test setup prior to the execution of the test, i.e. install the related application on the mobile and do any further actions required to run the test.
- shall provide results of the tests.
- shall check that the correct C-APDU is sent by the terminal on the interface with the Secure Element/ UICC / UICC Simulator (as specified in the ISO Command Expectation column).
- shall check that the correct R-APDU is received by the mobile application as the return value to the transmit() method (as specified in the API Expectation column).
- may check the R-APDU sent on the Secure Element/ UICC / UICC Simulator interface.
- should be able to automatically execute the tests.

5.3 Test format

5.3.1 Conformance requirements

The conformance requirements are expressed in the following way:

- Method prototype as listed in Open Mobile API specification.
- Normal execution:
  - Contains normal execution and correct parameters limit values, each referenced as a Conformance Requirement Normal (CRN).
- Parameters error:
  - Contains parameter errors and incorrect parameter limit values, each referenced as a Conformance Requirement Parameter Error (CRP).
- Context error:
  - Contains errors due to the context the method is used in, each referenced as a Conformance Requirement Context Error (CRC).

5.3.2 Initial conditions

In addition to the general preconditions defined in clause 5.4, this clause defines the initial conditions prior to the execution of each test case; i.e. for each ID.

5.3.3 Test procedure

Each test procedure contains a table of a number of test cases, each of these tests specified as follows:

Test case					
ID	**API Description**	**ISO Command Expectation DUT → UICC Simulator / SE**	**ISO Response UICC Simulator / SE → DUT**	**API Expectation**	**CRR**
The ID of the test case.	The name of the OMAPI method called by the test application.	The expected ISO command (C-APDU) received by the UICC Simulator / SE. It is sent by the DUT to UICC Simulator / SE as a result of the OMAPI method call.	The ISO response (R-APDU) sent by UICC Simulator / SE to the DUT as a response to the received ISO command.	The expected result of the OMAPI method called. E.g.: 'true' is returned.	The list of the Conformity Requirements which is the scope of the test case.

General notes regarding the ISO Command Expectation and ISO Response columns:

- Test cases test the implementation of the SIMalliance Open Mobile API implementation and not the behaviour of the Secure Elements. However to make sure the API is correctly implemented by the device, test cases verify command exchanges between the device and the SE/UICC Simulator as well as data and the result provided by API methods.
- The ISO Command Expectation is checked to validate if the OMAPI implementation sends the expected commands to the SE/UICC Simulator.
- The ISO Response is provided for information on the UICC Simulator /SE behaviour.
- The test procedure description contains APDUs. The TPDUs are not in the scope of the test specification so they are not listed in the test procedure descriptions.

- The APDUs exchanged during the access control procedure are out of scope of the test procedure description and shall be not considered as ISO command expectation or ISO response.
- Except for specific test cases aimed at checking the correct behaviour of the underlying transport protocol, all test cases are protocol agnostic.

Meaning of “No APDU”, “none” and “No selection” is defined in Chapter 1: Terminology.

5.4 General Initial Conditions

The General Initial Conditions are a set of general prerequisites prior to the execution of testing. The following rules apply:

- DUT shall be restarted for each test case and shall be ready for test execution.
- The test application is installed on the DUT.
- The test applets are installed on the SE.
- No logical channels must be open before execution of the test cases if not explicitly mentioned in the initial condition of the test case.

When working with UICC, DUT should not be connected to Telecom network to avoid unexpected APDU commands.

5.5 Mobile application and test controller

Unless otherwise specified, the test application shall be installed on the DUT.
The mobile application and the test controller are expected to be provided by test tool vendors.
SIMalliance provides a simple test runner as Android application. This application can execute the test cases and log results on the mobile. This test runner is not meant for compliance testing. It is provided as binary (APK) on SIMalliance web page.

5.6 Test case implementation

SIMalliance provides an implementation of the test cases in XML format. These test cases will be used by the SIMalliance test runner application and can be used by test tool manufacturers as a reference for certification. The test tool vendors are not required to use these XML files.
The XML files will be available on SIMalliance web page.

5.7 Secure Element Test applets

Unless otherwise specified, the required test applets shall be installed on the SE simultaneously.
A reference of these test applets will be available on SIMalliance web page (binary files) for download.

The following AIDs are used in the present document:

AID_TestApp	A0 00 00 06 00 01 00 01 EE 05 01
AID_TestApp_SW6999	A0 00 00 06 00 01 00 01 EE 05 02
AID_TestApp_SW6280	A0 00 00 06 00 01 00 01 EE 05 03
AID_TestApp_SW6283	A0 00 00 06 00 01 00 01 EE 05 04
AID_TestApp_SW6310	A0 00 00 06 00 01 00 01 EE 05 05
AID_TestApp_SW63C1	A0 00 00 06 00 01 00 01 EE 05 06

AID_TestApp_selectresponse	A0 00 00 06 00 01 00 01 EE 05 07
AID_TestApp_SW6280_selectresponse	A0 00 00 06 00 01 00 01 EE 05 08
AID_TestApp_SW6283_selectresponse	A0 00 00 06 00 01 00 01 EE 05 09
AID_TestApp_SW6310_selectresponse	A0 00 00 06 00 01 00 01 EE 05 0A
AID_TestApp_SW63C1_selectresponse	A0 00 00 06 00 01 00 01 EE 05 0B
AID_TestApp_p1p2	A0 00 00 06 00 01 00 01 EE 05 0C
AID_TestApp_clains	A0 00 00 06 00 01 00 01 EE 05 0D
AID_Partial_1	A0 00 00 06 00 01 00 01 EE 05 0E
AID_Partial_1_instance_1	<AID_Partial_1> 01
AID_Partial_1_instance_2	<AID_Partial_1> 02
AID_Partial_2	<AID_Partial_1_instance_1>
AID_Partial_2_instance_1	<AID_Partial_2>
AID_Partial_SW6280	A0 00 00 06 00 01 00 01 EE 05 0F
AID_Partial_SW6280_instance_1	<AID_Partial_SW6280> 01
AID_Partial_SW6280_instance_2	<AID_Partial_SW6280> 02
AID_Partial_SW6283	A0 00 00 06 00 01 00 01 EE 05 10
AID_TestApp_SW61xx	A0 00 00 06 00 01 00 01 EE 05 11
AID_Partial_SW6283_instance_1	<AID_Partial_SW6283> 01
AID_Partial_SW6283_instance_2	<AID_Partial_SW6283> 02
AID_TestApp_multiselecteable	A0 00 00 06 00 01 00 01 EE 55 01
AID_accessdenied	A0 00 00 06 00 01 00 01 EE 05 FE
AID_nonexisting	A0 00 00 06 00 01 00 01 EE 05 FF
AID_illegal_1	A0 00 00 06
AID_illegal_2	A0 00 00 06 00 01 00 01 EE 10 00 10 00 60 00 00 0A
AID_TestApp_Multi_SW61xx	A0 00 00 06 00 01 00 01 EE 05 12
AID_TestApp_Get_Response	A0 00 00 06 00 01 00 01 EE 05 13

Table 7: Used AIDs

5.7.1 Test APDU Interface

This table gives the list of commands that are used in test cases and that are supported by the Secure Element Test applets.
The values for “Cla” depend on the test case: in most of the test cases the Cla contains a logical channel number.

	Cla	Ins	P1	P2	Lc	Data	Le
Test_APDU1	0x	10 (case 4)	01 (for echo of the payload)	00	length	Data	00
Test_APDU2	0x	10 (case 4)	02 (echo of the payload	00	length	Data	00

			with long delay (more than 1 sec) before return)				
Test_APDU3	0x	20 (filtered APDU)	00	00	length	Data	00
Test_APDU4	0x	30 (case 1)	00	00			
Test_APDU5	0x	40 (case 2)	00	00			00
Test_APDU6	0x	50 (case 3)	00	00	length	Data	
Test_APDU7	0x	55 case1	00 (waiting time extension has to be send)	00			
Test_APDU8	0x	40	00	00			04
APDU_case1	0x	01	01-32	00			
APDU_case2	0x	02	01-11	00			FF
	0x	02	12-32	00			
APDU_case3	0x	03	01-32	00	FF	Data	
APDU_case4	0x	04	01-11	00	FF	Data	FF
	0x	04	12-32	00	FF	Data	
APDU	00-FE	00-FF excluding: 0x70, 0x6x,	10	00	10	Data	10

		0x9x					
APDU_MANAGE_CH_OPEN	0x	70	00	00			01
APDU_MANAGE_CH_CLOSE	0x	70	80	01			
APDU_SELECT_BY_FID	0x	A4	00	00	02	3F00	00 or empty
APDU_SELECT_BY_DF	0x	A4	04	00	XX	'AID'	00 or empty
APDU_GET_RESPONSE	0x	C0	00	00			04
APDU_LONG_RESPONSE	0x	60(case 2)	00	00			00

Table 8: List of APDU Commands for Test Applets

For some test cases, APDU Status Words (SW1-SW2) values depend on P1 value of the C-APDU (only for APDU_case1, APDU_case2, APDU_case3, APDU_case4):

P1	**SW1-SW2**
0x01	0x6200
0x02	0x6202
0x03	0x6280
0x04	0x6281
0x05	0x6282

0x06	0x6283
0x07	0x6284
0x08	0x6285
0x09	0x6286
0x0A	0x62F1
0x0B	0x62F2
0x0C	0x6300
0x0D	0x6381
0x0E	0x63C2
0x0F	0x6310
0x10	0x63F1
0x11	0x63F2
0x12	0x6400
0x13	0x6401
0x14	0x6402
0x15	0x6480
0x16	0x6500

0x17	0x6581
0x18	0x6800
0x19	0x6881
0x1A	0x6882
0x1B	0x6883
0x1C	0x6884
0x1D	0x6900
0x1E	0x6900
0x1F	0x6981
0x20	0x6982
0x21	0x6983
0x22	0x6984
0x23	0x6985
0x24	0x6986
0x25	0x6987
0x26	0x6988
0x27	0x6A00

0x28	0x6A80
0x29	0x6A81
0x2A	0x6A82
0x2B	0x6A83
0x2C	0x6A84
0x2D	0x6A85
0x2E	0x6A86
0x2F	0x6A87
0x30	0x6A88
0x31	0x6A89
0x32	0x6A8A

Table 9: P1 - Status Word Pairs

The length of the data and the data bytes may be adapted by the test controller for different test runs (e.g. run the test cases with different data length during different test runs). The test applet must be able to handle different data length.

5.8 Access Control Configuration

To test security errors two rules shall be defined complying with GP SEAC:

- Rule 1: It denies access to AID_accessdenied from any mobile application.
- Rule 2: Denies sending a specific APDU command: Test_APDU3 to AID_TestApp from any mobile application.

For all other tests, a rule granting access to all applets for all mobile applications shall be used.
An example of ARA applet and ARF configuration is provided in Annex B.

6. Test Cases

6.1 Class SEService

The SEService realizes the communication to available Secure Elements on the device.
This is the entry point of this API. It is used to connect to the infrastructure and get access to a list of Secure Element Readers.

6.1.1 Constructor: SEService(Context context, SEService.CallBack listener)

(a) Conformance Requirements

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

```
SEService(Context context, SEService.CallBack listener)
```

Normal execution
CRN1: Establishes a new connection that can be used to connect to all the Secure Elements available in the DUT.
CRN2: The isConnected() method returns true after the connection process is finished.
CRN3: The serviceConnected() method of the listener object is called.

Parameter errors
CRP1: IllegalParameterError or NullPointerError – if the parameter “context” is null.

Context errors
None

(b) Initial Conditions

(c) Test Procedure

Test case					
ID	**API Description**	**ISO Command Expectation DUT → UICC Simulator / SE**	**ISO Response UICC Simulator / SE → DUT**	**API Expectation**	**CRR**
1	**SEService Constructor with 2 Parameters**				
	Constructor: SEService(context, listener)	none	none	serviceConnected() method of the listener object is called (recommended: within 10 sec).	CRN1 CRN3
2	**SEService Constructor and check with isConnected**				
	1. Constructor: SEService(context, listener) 2. After seService.serviceConnected() callback received; seService.isConnecte	none	none	2. seService.isConnected() returns true	CRN2

	d()				
3	**SEService Constructor with missing Context**				
	Constructor: SEService(null, listener)	none	none	`IllegalParameterError or NullPointerError expected`	CRP1
4	**SEService Constructor with missing Listener**				
	1. Constructor: SEService(context, null) *2.* *-- wait 10 sec (not blocking) --* seService.isConnected()	none	none	2. seService.`isConnected() returns true`	CRP1
5	**SEService Constructor without any parameters**				
	Constructor: SEService(null, null)	none	none	`IllegalParameterError or NullPointerError expected`	CRP1
6	Use of a second SEService instance				
	1. Constructor: SEService(context, listener) 2. After seService.serviceConnected() callback received; seService.isConnected() 3. create a second SEService object Constructor: seService2 = SEService(context, listener) 4. After seService2.serviceConnected() callback received; seService2.isConnected()	none	none	2. seService.isConnected() returns true 4. seService2.isConnected() returns true	CRN2

6.1.2 Method: Reader[] getReaders()

(a) Conformance Requirements

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

```
Reader[] getReaders()
```

Normal execution
CRN1: Reader[] contains the list of available secure element readers.
CRN2: If there is no reader, then the array of readers returned by getReaders() method has length 0.
CRN3: There must be no duplicated objects in the list of readers.

Parameter errors
None

Context errors
None

(b) Initial Conditions

SEService Object has been created and the isConnected() method has been called and has returned true.

(c) Test Procedure

Test case					
ID	**API Description**	**ISO Command Expectation DUT → UICC Simulator / SE**	**ISO Response UICC Simulator / SE → DUT**	**API Expectation**	**CRR**
1	**SEService GetReaders with return of multiple readers**				
	seService.getReaders ()	None	None	Returned array contains list with the correct number of the supported readers ; There must be no duplicated entries in the list	CRN1 CRN3

6.1.3 Method: boolean isConnected ()

(a) Conformance Requirements

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

```
boolean isConnected ()
```

Normal execution
CRN1: isConnected() returns true if the service is connected.
CRN2: isConnected() returns false if the service is not connected.

Parameter errors
None

Context errors
None

(b) Initial Conditions

SEService Object has been created and the isConnected() method has been called and has returned true.

(c) Test Procedure

Test case					
ID	**API Description**	**ISO Command Expectation DUT → UICC Simulator / SE**	**ISO Response UICC Simulator / SE → DUT**	**API Expectation**	**CRR**
1	**SEService isConnected returns true**				
	seService.isConnected()	none	none	Returns true	CRN1
2	**SEService isConnected return false**				
	1. seService.shutdown() 2. seService.isConnected()	none	none	2. Returns false	CRN2

6.1.4 Method: void shutdown ()

(a) Conformance Requirements

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

```
Void shutdown ()
```

Normal execution
CRN1: Releases all Secure Elements resources allocated by this SEService.
CRN2: As a result isConnected() will return false after shutdown() was called.
CRN3: After this method call, the state of SEService object is invalid (not connected any more).

Parameter errors
None

Context errors
None

(b) Initial Conditions

SEService Object has been created and the isConnected() method has been called and has returned true.

(c) Test Procedure

Test case					
ID	**API Description**	**ISO Command Expectation DUT → UICC Simulator / SE**	**ISO Response UICC Simulator / SE → DUT**	**API Expectation**	**CRR**
1	**SEService shutdown with no channels open**				
	1. seService.shutdown() 2. seService.isConnected() 3. seService.getReaders()	none	none	2. seService.isConnected returns false 3. IllegalStateError	CRN1 CRN2 CRN3

2	**SEService shutdown with one channel open**				
	1. seService.getReaders() 2. reader.openSession(firstReader) 3. session.openLogicalChannel(AID_TestApp) 4. seService.shutdown() 5. seService.isConnected()	 CMD 3-1: APDU_MANAGE_CH_OPEN CMD 3-2: APDU_SELECT_BY_DF – CLA contains the Channel Number returned by the card in RESP 3-1; Data = 'AID_TestApp' CMD-4-1: MANAGE CHANNEL (P1='80')	 RESP 3-1: R-APDU - Data: Channel Number; SW '90 00' RESP 3-2: R-APDU - SW '90 00' RESP 4-1: R-APDU - SW '90 00'	no errors or exceptions are expected 5. seService.isConnected returns false	CRN1 CRN2
3	**SEService shutdown during transmit in different thread**				
	1. seService.getReaders() 2. reader.openSeesion(firstReader) 3. session.openLogicalChannel(AID_TestApp) *4.* *-- Start new thread – Channel.transmit(***Test_APDU2** *5.* *-- return to first thread right after transmit returned the response --* seService.shutdown() 6. seService.isConnected()	 CMD 3-1: APDU_MANAGE_CH_OPEN CMD 3-2: APDU_SELECT_BY_DF – CLA contains the Channel Number returned by the card in RESP 3-1; Data = 'AID_TestApp' CMD 4-1: C-APDU ('XX 10 02 00 04 01 02 03 04 00') CMD 5-1: MANAGE CHANNEL (P1='80')	 RESP 3-1: R-APDU - Data: Channel Number; SW '90 00' RESP 3-2: R-APDU - SW '90 00' RESP 4-1: R-APDU – '01 02 03 04' SW '90 00' RESP 5-1: R-APDU - SW '90 00'	no errors or exceptions are expected 4. byte[ ]= {'01, 02, 03, 04, 90, 00} (transmit executed successfully) 6. seService.isConnected returns false	CRN1 CRN2

6.1.5 Method: void getVersion()

(a) Conformance Requirements

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

```
String getVersion()
```

Normal execution
CRN1: Returns the version of the OpenMobile API Specification this implementation is based on.

Parameter errors
None

Context errors
None

(b) Initial Conditions

SEService Object has been created and the isConnected() method has been called and has returned true.

(c) Test Procedure

Test case					
ID	**API Description**	**ISO Command Expectation DUT → UICC Simulator / SE**	**ISO Response UICC Simulator / SE → DUT**	**API Expectation**	**CRR**
1	**getVersion returns version string**				
	1. seService.getVersion()	none	none	1. returns a String that contains the OpenMobile API version (e.g. 2.05)	CRN1

6.2 Class (or interface): SEService.CallBack

Interface to receive call-backs when the service is connected.
If the target language and environment allows it, then this shall be an inner interface of the SEService class.

6.2.1 Method: void serviceConnected(SEService service)

(a) Conformance Requirements

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

```
void serviceConnected(SEService service)
```

Normal execution
CRN1: The SEService object parameter must be the object that was created as result of the SEService constructor and must not be null.

Parameter errors
None

Context errors
None

(b) Initial Conditions

SEService Constructor called

(c) Test Procedure

Test case					
ID	**API Description**	**ISO Command Expectation DUT → UICC Simulator / SE**	**ISO Response UICC Simulator / SE → DUT**	**API Expectation**	**CRR**
1	**SEService Callback received after constructor**				
	1. serviceConnected(SEService service) received; 2. Call seService.isConnected() of received SEService object	none	none	1. SEService object created with constructor and the one received in the callback are the same object 2. seService .isConnected returns true.	CRN1

6.3 Class Reader

The instances of this class represent Secure Element readers connected to this device. These readers can be physical devices or virtual devices. They can be removable or not. They can contain one Secure Element that can or cannot be removed.

6.3.1 Method: String getName()

(a) Conformance Requirements

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

```
String getName()
```

Normal execution

CRN1: Return the name of this reader.

- If this reader is a SIM reader, then its name must be "SIM[Slot]".
- If the reader is a SD or micro SD reader, then its name must be "SD[slot]".
- If the reader is an embedded SE reader, then its name must be "eSE[slot]".

Slot is a decimal number without leading zeros. The numbering must start with 1 (e.g. SIM1, SIM2, … or SD1, SD2, … or eSE1, eSE2, …). The slot number "1" for a reader is optional (SIM and SIM1 are both valid for the first SIM-reader, but if there are two readers then the second reader must be named SIM2). This applies also for other SD or SE readers.

Context errors

None

(b) Initial Conditions

SEService Object has been created and the isConnected() method has been called and has returned true.

The “reader” instance is an element of a Reader[] array, returned by invoking seService.getReaders() method.

(c) Test Procedure

Test case					
ID	**API Description**	**ISO Command Expectation DUT → UICC Simulator / SE**	**ISO Response UICC Simulator / SE → DUT**	**API Expectation**	**CRR**
1	Get Name				
	reader.getName()	none	none	Returned String is not null and returns the correct string. E.g.: “SIM1 or SIM” for the first SIM reader. No exception is expected.	CRN1

6.3.2 Method: SEService getSEService()

(a) Conformance Requirements

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

```
SEService getSEService()
```

Normal execution
CRN1: Get the SEService that provides this Reader

Parameter errors
None

Context errors
None

(b) Initial Conditions

SEService Object has been created and the isConnected() method has been called and has returned true.
The “reader” instance is an element of a Reader[] array, returned by invoking seService.getReaders() method.

(c) Test Procedure

Test case					
ID	**API Description**	**ISO Command Expectation DUT → UICC Simulator / SE**	**ISO Response UICC Simulator / SE→DUT**	**API Expectation**	**CRR**
1	Get SEService and compare				
	reader.getSEService ()== service	None	None	No exception is expected (SEService object is not null and is the same SEService	CRN1

				object which provides this Reader.) .	

6.3.3 Method: boolean isSecureElementPresent()

(a) Conformance Requirements

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

```
boolean isSecureElementPresent()
```

Normal execution

CRN1: This method checks if a Secure Element is present in the reader, in case of its presence it returns true.

CRN2: This method returns false if the Secure Element is not present in the reader.

Parameter errors

None

Context errors

None

(b) Initial Conditions

SEService Object has been created and the isConnected() method has been called and has returned true.

The "reader" instance is an element of a Reader[] array, returned by invoking seService.getReaders() method.

Test case ID1: The SE used for testing is available and accessible.

Test case ID2: The SE that is tested is not inserted.

(c) Test Procedure

Test case					
ID	**API Description**	**ISO Command Expectation DUT →UICC Simulator / SE**	**ISO Response UICC Simulator / SE → DUT**	**API Expectation**	**CRR**
1	Secure Element is present				
	reader.isSecureElementPresent()	None	None	True is returned No exception is expected.	CRN1
2	Secure Element is not present				
	reader.isSecureElementPresent()	None	None	False is returned No exception is expected.	CRN2

6.3.4 Method: Session openSession()

(a) Conformance Requirements

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

```
Session openSession()
```

Normal execution
CRN1: This method allows an application to connect to a Secure Element in the reader.
CRN2: The Secure Element needs to be prepared (initialized) for communication (i.e. switched on).
CRN3: There might be multiple sessions opened at the same time on the same reader.
CRN4: This method returns a Session object to be used to create Channels.

Parameter errors
None

Context errors
CRC1: IOError - something went wrong with the communication to the Secure Element.

(b) Initial Conditions

SEService Object has been created and the isConnected() method has been called and has returned true.
The "reader" instance is an element of a Reader[] array, returned by invoking seService.getReaders() method.
Test case ID1: A SE is connected to the Reader. No opened Sessions.
Test case ID2: A SE is connected to the Reader.

(c) Test Procedure

Test case					
ID	**API Description**	**ISO Command Expectation DUT →UICC Simulator / SE**	**ISO Response UICC Simulator / SE → DUT**	**API Expectation**	**CRR**
1	First Session opening				
	reader.openSession ()	None	None	Returned Session object is not null. No exception is expected	CRN1 CRN2 CRN4
2	Second Session opening				
	1. Session s1 = reader.openSession (); **2. Session s2 = reader.openSession ();** **3. s1 != s2;**	None	None	1. No exception is expected. 2. No exception is expected. 3.Session instances s1 and s2 are not the same. No exception is expected.	CRN1 CRN2 CRN3 CRN4

6.3.5 Method: void closeSessions()

(a) Conformance Requirements

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

```
void closeSessions()
```

Normal execution
CRN1: This method closes all the sessions opened on this reader.
CRN2: All the channels opened by all this session will be closed.

Parameter errors
None

Context errors
None

(b) Initial Conditions

SEService Object has been created and the isConnected() method has been called and has returned true.
The "reader" instance is an element of a Reader[] array, returned by invoking seService.getReaders() method.
Test case ID1: A SE is connected to the Reader. Session instances s1 and s2 are created.
Test case ID2: A SE is connected to the Reader. Session instance s1 is created. Three logical channels are opened within 's1'.

(c) Test Procedure

Test case					
ID	**API Description**	**ISO Command Expectation DUT → UICC Simulator / SE**	**ISO Response UICC Simulator / SE → DUT**	**API Expectation**	**CRR**
1	Close sessions				
	1. reader.closeSessions() **2. s1.isClosed();** **3. s2.isClosed();**	None	None	1. No exception is expected 2. return 'true' 3. return 'true'	CRN1
2	Close sessions and channels				
	reader.closeSessions();	CMD 1-1: MANAGE CHANNEL (P1='80') CMD 1-2: MANAGE CHANNEL (P1='80') CMD 1-3: MANAGE CHANNEL (P1='80')	RESP 1-1: R-APDU - SW '90 00' RESP 1-2: R-APDU - SW '90 00' RESP 1-3: R-APDU - SW '90 00'	No exception is expected.	CRN2

6.4 Class Session

6.4.1 Method: Reader getReader()

(a) Conformance Requirements

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

```
Reader getReader()
```

Normal execution
CRN1: Get the reader that provides this session.

Parameter errors
None

Context errors
None

(b) Initial Conditions

SEService Object has been created and the isConnected() method has been called and has returned true.
A Reader instance "reader" is selected and a Session instance "session" is opened with the selected "reader".

(c) Test Procedure

Test case					
ID	**API Description**	**ISO Command Expectation DUT -> UICC Simulator / SE**	**ISO Response UICC Simulator / SE -> DUT**	**API Expectation**	**CRR**
1	**Return the Reader object for a Session instance**				
	session.getReader()	None.	None.	Returned Reader object is not null. No exception is expected.	CRN1
2	**Get the Reader object and compare with the object that provides this session**				
	session.getReader() ==reader	None.	None.	The Reader object returned by getReader() is the same object as the one which provides this session. No exception is expected.	CRN1

6.4.2 Method: byte[] getATR()

(a) Conformance Requirements

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

```
byte[] getATR()
```

Normal execution
CRN1: This method gets the Answer to Reset of this Secure Element.
CRN2: If the ATR for this Secure Element is not available the returned byte array is null.

Parameter errors
None

Context errors
None

(b) Initial Conditions

SEService Object has been created and the isConnected() method has been called and has returned true.
A Reader instance "reader" is selected and a Session instance "session" is opened with the selected "reader".

Test case ID1 and ID2: The UICC Simulator / SE has sent its "ATR" to the DUT.

(c) Test Procedure

Test case					
ID	**API Description**	**ISO Command Expectation DUT -> UICC Simulator / SE**	**ISO Response UICC Simulator / SE -> DUT**	**API Expectation**	**CRR**
1	**Return the Answer To Reset**				
	session.getATR();	None	None	No exception is expected.	CRN1
2	**Returned Answer To Reset equals to the "ATR" sent during reset**				
	session.getATR()== ATR;	None	None	The Answer to Reset returned by getATR() equals to the "ATR" sent by the UICC Simulator / SE. No exception is expected.	CRN1
3	**Return null in case the Answer To Reset is not available**				
	session.getATR();	None	None	Null is expected to return. No exception is expected.	CRN2

6.4.3 Method: void close()

(a) Conformance Requirements

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

```
void close()
```

Normal execution
CRN1: Close the connection with the Secure Element.
CRN2: This API will close any channels opened by this application with this Secure Element.

Parameter errors
None

Context errors
None

(b) Initial Conditions

SEService Object has been created and the isConnected() method has been called and has returned true.
A Reader instance "reader" is selected and a Session instance "session" is opened with the selected "reader".
Test case ID1: One logical channel is opened to AID_TestApp.
Test case ID2: Three logical channels are opened to AID_TestApp_multiselectable.

(c) Test Procedure

Test case					
ID	**API Description**	**ISO Command Expectation DUT -> UICC Simulator / SE**	**ISO Response UICC Simulator / SE -> DUT**	**API Expectation**	**CRR**
1	**Close a session and check the state**				
	session.close();	MANAGE CHANNEL (P1='80')	R-APDU - SW '90 00'	No exception is expected.	CRN1
2	**Close a session with more logical channels**				
	1. session.close();	CMD 1-1: MANAGE CHANNEL (P1='80') CMD 1-2: MANAGE CHANNEL (P1='80') CMD 1-3: MANAGE CHANNEL (P1='80')	RESP 1-1: R-APDU - SW '90 00' RESP 1-2: R-APDU - SW '90 00' RESP 1-3: R-APDU - SW '90 00'	1. No exception is expected.	CRN2

6.4.4 Method: boolean isClosed()

(a) Conformance Requirements

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

```
boolean isClosed()
```

Normal execution
CRN1: Tells if this session is closed: if so, isClosed returns “true”.
CRN2: If the session is open it returns false.

Parameter errors
None

Context errors
None

(b) Initial Conditions

SEService Object has been created and the isConnected() method has been called and has returned true.
A Reader instance "reader" is selected and a Session instance "session" is opened with the selected “reader”.

(c) Test Procedure

Test case					
ID	**API Description**	**ISO Command Expectation DUT -> UICC Simulator / SE**	**ISO Response UICC Simulator / SE -> DUT**	**API Expectation**	**CRR**
1	**Check a session already closed**				
	1. session.close(); **2. session.isClosed();**	None	None	1. No exception is expected. 2. “true” is expected to return. No exception is expected.	CRN1
2	**Check an open session**				
	session.isClosed();	None	None	“false” is expected to return. No exception is expected.	CRN2

6.4.5 Method: void closeChannels()

(a) Conformance Requirements

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

```
void closeChannels()
```

Normal execution
CRN1: Close any channel opened on this session.

Parameter errors
None

Context errors
None

(b) Initial Conditions

SEService Object has been created and the isConnected() method has been called and has returned true.
A Reader instance "reader" is selected and a Session instance "session" is opened with the selected "reader".
Test case ID1: Three logical channels are opened to AID_TestApp_multiselectable.
Test case ID2: No logical channel is opened.

(c) Test Procedure

Test case					
ID	**API Description**	**ISO Command Expectation DUT -> UICC Simulator / SE**	**ISO Response UICC Simulator / SE -> DUT**	**API Expectation**	**CRR**
1	**Close all the channels opened by the session**				
	1. session.closeChannels();	CMD 1-1: MANAGE CHANNEL (P1='80') CMD 1-2: MANAGE CHANNEL (P1='80') CMD 1-3: MANAGE CHANNEL (P1='80')	RESP 1-1: R-APDU - SW '90 00' RESP 1-2: R-APDU - SW '90 00' RESP 1-3: R-APDU - SW '90 00'	1. No exception is expected.	CRN1
2	**Close if no channel is open**				
	1. session.closeChannels();	No APDU	None	1. No exception is expected.	CRN1

6.4.6 Method: Channel openBasicChannel(byte[] aid)

(a) Conformance Requirements

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

```
Channel openBasicChannel(byte[] aid)
```

Normal execution
CRN1: Get an access to the basic channel, as defined in the ISO7816-4 specification (the one that has number 0). The obtained object is an instance of the Channel class.
CRN2: The AID can be null, which means no SE application is to be selected on this channel and the default SE application is used. If the default SE application is not currently selected on the basic channel then null will be returned.

CRN3: Once this channel has been opened by a device application, it is considered as "locked" by this device application, and other calls to this method will return null, until the channel is closed.
CRN4: Returns null, if the basic channel is locked (e.g. by the Secure Element drivers).

Parameter errors
CRP1: IllegalParameterError - if the aid's length is not within 5 to 16 (inclusive).

Context errors
CRC1: IOError - if something goes wrong with the communication to the reader or the Secure Element.
CRC2: NoSuchElementError – If the AID on the Secure Element is not available.
CRC3: IllegalStateError - if the Secure Element session is used after being closed.
CRC4: SecurityError - if the calling application cannot be granted access to this AID on this session.

(b) Initial Conditions

SEService Object has been created and the isConnected() method has been called and has returned true.
A Reader instance "reader" is selected and a Session instance "session" is opened with the selected "reader".
Test case ID4a: AID_TestApp is installed as the default selected applet.
Test case ID4b: The default applet is different from the AID_TestApp, e.g. USIM on a UICC SE, etc.

(c) Test Procedure

Test case					
ID	**API Description**	**ISO Command Expectation DUT -> UICC Simulator / SE**	**ISO Response UICC Simulator / SE -> DUT**	**API Expectation**	**CRR**
1	**Open a basic channel**				
	1. session.openBasicChannel (AID_TestApp);	CMD 1: APDU_SELECT_BY_DF; Data = 'AID_TestApp'	RESP 1: R-APDU - SW '90 00'	1. Returned Channel object is not null. No exception is expected.	CRN1
2	**Open a basic channel and check, if the selected SE applet answers**				
	1. session.openBasicChannel (AID_TestApp);	CMD 1: APDU_SELECT_BY_DF; Data = 'AID_TestApp'	RESP 1: R-APDU - SW '90 00'	1. Returned Channel object is not null. No exception is expected.	CRN1
	2. channel.transmit(Test_APDU1)	CMD 2: C-APDU ('00 10 01 00 04 01 02 03 04 00')	RESP 2: R-APDU - Data = '01 02 03 04'; SW '90 00'	2. Returned Response equals to 'R-APDU' - Data = ' 01 02 03 04'; SW '90 00'. No exception is expected.	
3	**Open a basic channel with the default SE applet**				
	1. session.openBasicChannel (null);	no selection <on basic channel>	None	1. Returned Channel object is not null. No exception is expected.	CRN2
4a	**Open a basic channel with AID_TestApp as the default SE applet and check, if the applet answers**				
	1. session.openBasicChannel (null);	no selection <on basic channel>	None	1. Returned Channel object is not null. No exception is expected.	CRN2 CRN3

	2. channel.transmit(Test_APDU1); **3. session.openBasicChannel (null);**	CMD 2: C-APDU (’00 10 01 00 04 01 02 03 04 00’) no APDU	RESP 2: R-APDU - Data = ’01 02 03 04’; SW ‘90 00’ None	2. Returned Response equals to ‘R-APDU’ - Data = ‘01 02 03 04’; SW ‘90 00’. 3 Returned Channel object is null. No exception is expected.	
4b	**Open a basic channel with the default SE applet and check, if the applet answers**				
	1. session.openBasicChannel (null); **2. channel.transmit(Test_APDU1);** **3. session.openBasicChannel (null);**	no selection <on basic channel> CMD 2: C-APDU (’00 10 01 00 04 01 02 03 04 00’) No APDU	None RESP 2: SW ‘6D 00’ or SW ‘6E 00’ None	1. Returned Channel object is not null. No exception is expected. 2. Returned SW ‘6D 00’ or SW ‘6E 00’ No exception is expected. 3 Returned Channel object is null. No exception is expected.	CRN2 CRN3
5	**Open a basic channel with the default SE applet when the default applet is not currently selectable**				
	1. session.openBasicChannel (AID_TestApp); **2. channel.close();** **3. session.openBasicChannel (null);**	CMD 1: APDU_SELECT_BY_DF; Data = ‘AID_TestApp‘ CMD 2: None CMD 3: No APDU	RESP 1: R-APDU - SW ‘90 00’ RESP 2: None RESP 3: None	1. Returned Channel object is not null. No exception is expected 2. No exception is expected 3. Returned Channel object is null. No exception is expected.	CRN2
6	**Open a basic channel when it is locked by an application**				
	1. session.openBasicChannel (AID_TestApp); **2. session.openBasicChannel (AID_TestApp_multiselectable);**	CMD 1: APDU_SELECT_BY_DF; Data = ‘AID_TestApp‘ CMD 2: No APDU command is expected. (the channel is locked by the API)	RESP 1: R-APDU - SW ‘90 00’ RESP 2: No Response.	1. Returned Channel object is not null. No exception is expected. 2. Returned Channel object is null. No exception is expected.	CRN3
7	**Open a basic channel when it is locked by default**				
	session.openBasicChannel (AID_TestApp);	No APDU	None.	Returned Channel object is null. No exception is expected.	CRN4
8	**The length of the AID is less than 5**				

	session.openBasicChannel (AID_Illegal_1);	No APDU	None	IllegalParameterError is expected.	CRP1
9	**The length of the AID is more than 16**				
	session.openBasicChannel (AID_Illegal_2);	No APDU	None	IllegalParameterError is expected.	CRP1
10	**Communication problem with the Secure Element**				
	session.openBasicChannel (AID_TestApp);	APDU_SELECT_BY_DF; Data = 'AID_TestApp'	No R-APDU is returned.	IOError is expected.	CRC1
11	**The AID is not available on the Secure Element**				
	session.openBasicChannel (AID_nonexisting);	APDU_SELECT_BY_DF; Data = 'AID_nonexisting '	R-APDU – SW '6A 82'	NoSuchElementError is expected.	CRC2
12	**Open a basic channel, when session is already closed**				
	1. session.close(); **2. session.openBasicChannel (AID_TestApp);**	None No APDU	None	1. No exception is expected. 2. IllegalStateError is expected.	CRC3
13	**The application opening the basic channel has no access to the selected SE applet**				
	session.openBasicChannel (AID_accessdenied);	no selection <for AID_accessdenied>	None.	SecurityError is expected.	CRC4

6.4.7 Method: Channel openLogicalChannel(byte[] aid)

(a) Conformance Requirements

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

```
Channel openLogicalChannel(byte[] aid)
```

Normal execution

CRN1: Open a logical channel with the Secure Element, selecting the application represented by the given AID.

CRN2: If the AID is null, then the default application shall be used.

CRN3: It's up to the Secure Element to choose which logical channel will be used.

CRN4: Return null if Secure Element is unable to provide a new logical channel.

CRN5: If the selection of the SE applet fails the logical channel shall be closed.

CRN6: If the status word indicates that the Secure Element was able to open a channel (e.g. status word '90 00' or status words referencing a warning in ISO-7816-4: '62 XX" or "63 XX') the API shall keep the channel opened.

CRN7: If the device is forbidding using a null AID.

Parameter errors

CRP1: IllegalParameterError - if the aid's length is not within 5 to 16 (inclusive).

Context errors

CRC1: IOError - if something goes wrong with the communication to the reader or the Secure Element. (e.g. Secure Element is no longer available).

CRC2: NoSuchElementError - if the AID on the Secure Element is not available (or cannot be selected) or a logical channel is already open to a non-multiselectable applet.

CRC3: IllegalStateError - if the Secure Element session is used after being closed.

CRC4: SecurityError - if the calling application cannot be granted access to this AID on this session.

(b) Initial Conditions

SEService Object has been created and the isConnected() method has been called and has returned true.

A Reader instance "reader" is selected and a Session instance "session" is opened with the selected "reader".

Test case ID4a: AID_TestApp is installed as the default selected applet.

Test case ID4b: The default applet is different from the AID_TestApp, e.g. USIM on a UICC SE, etc.

Test case ID5: The maximum number of logical channels supported by the UICC Simulator / SE is already opened to AID_TestApp_multiselectable.

(c) Test Procedure

Test case					
ID	**API Description**	**ISO Command Expectation DUT -> UICC Simulator / SE**	**ISO Response UICC Simulator / SE -> DUT**	**API Expectation**	**CRR**
1	**Open a logical channel**				
	1. session.openLogicalChannel (AID_TestApp);	CMD 1-1: APDU_MANAGE_CH_OPEN CMD 1-2: APDU_SELECT_BY_DF – CLA contains the Channel Number returned by the card in RESP 1-1;; Data =	RESP 1-1: R-APDU - Data: Channel Number; SW '90 00' RESP 1-2: R-APDU - SW '90 00'	1. Returned Channel object is not null. No exception is expected.	CRN1 CRN3

		'AID_TestApp'			
2	**Open a logical channel and check, if the selected SE applet answers**				
	1. session.openLogicalChannel (AID_TestApp);	CMD 1-1: APDU_MANAGE_CH_OPEN CMD 1-2: APDU_SELECT_BY_DF – CLA contains the Channel Number returned by the card in RESP 1-1;; Data = 'AID_TestApp'	RESP 1-1: R-APDU - Data: Channel Number; SW '90 00' RESP 1-2: R-APDU - SW '90 00'	1. Returned Channel object is not null. No exception is expected.	CRN1
	2. channel.transmit(Test_APDU1)	CMD 2: C-APDU ('XX 10 01 00 04 01 02 03 04 00')	RESP 2: R-APDU - Data = '01 02 03 04'; SW '90 00'	2. Returned Response equals to 'R-APDU' - Data = '01 02 03 04'; SW '90 00'. No exception is expected.	
3a	**Open a logical channel with the default SE applet**				
	1. session.openLogicalChannel (null);	CMD 1: APDU_MANAGE_CH_OPEN	RESP 1: R-APDU - Data: Channel Number; SW '90 00'	1. Returned Channel object is not null. No exception is expected.	CRN2
3b	**Open a logical channel with the default SE applet when it is not supported by device**				
	1. session.openLogicalChannel (null);	No APDU	None.	1. Returned Channel object is null. No exception is expected.	CRN7
4a	**Open a logical channel with AID_TestApp as the default SE applet and check, if the applet answers**				
	1. session.openLogicalChannel (null);	CMD 1: APDU_MANAGE_CH_OPEN	RESP 1: R-APDU - Data: Channel Number; SW '90 00'	1. Returned Channel object is not null. No exception is expected.	CRN2
	2. channel.transmit(Test_APDU1);	CMD 2: C-APDU ('XX 10 01 00 04 01 02 03 04 00')	RESP 2: R-APDU - Data = '01 02 03 04'; SW '90 00'	2. Returned Response equals to 'R-APDU' - Data = '01 02 03 04 '; SW '90 00'.	
4b	**Open a logical channel with the default SE applet and check, if the applet answers**				
	1. session.openLogicalChannel (null);	CMD 1: APDU_MANAGE_CH_OPEN	RESP 1: R-APDU - Data: Channel Number; SW '90 00'	1. Returned Channel object is not null. No exception is expected.	CRN2
	2. channel.transmit(Test_APDU1);	CMD 2: C-APDU ('XX 10 01 00 04 01 02 03 04 00')	RESP 2: SW '6D 00' or SW '6E 00'	2. Returned SW '6D 00' or SW '6E 00' No exception is expected.	
5a	**Open a logical channel, when no new logical channel is available, device manages maximum number of logical channels**				
	1. session.openLogicalChannel (AID_TestApp);	No APDU	none	1. Returned Channel object is null. No exception is	CRN4

				expected.	
5b	**Open a logical channel, when no new logical channel is available, device does not manage maximum number of logical channels and access control is checked on openSession method**				
	1. session.openLogicalChannel (AID_TestApp);	CMD 1: APDU_MANAGE_CH_OPEN	RESP 1: R-APDU – SW '68 81' or '6A 81'	1. Returned Channel object is null. No exception is expected.	CRN4
5c	**Open a logical channel, when no new logical channel is available, device does not manage maximum number of logical channels and access control is checked on openLogicalChannel method**				
	1. session.openLogicalChannel (AID_TestApp);	no selection <for AID_TestApp>	None	SecurityError is expected.	CRN4
6	**The length of the AID is less than 5**				
	session.openLogicalChannel (AID_Illegal_1);	No APDU	None	IllegalParameterError is expected.	CRP1
7	**The length of the AID is more than 16**				
	session.openLogicalChannel (AID_Illegal_2);	No APDU	None	IllegalParameterError is expected.	CRP1
8	**Communication problem with the Secure Element**				
	1. session.openLogicalChannel (AID_TestApp);	CMD 1-1: APDU_MANAGE_CH_OPEN	None	1. IOError is expected. (APDU must not be resent automatically even if the SE/UICC Simulator answers after reset.)	CRC1
9	**The AID is not available on the Secure Element**				
	1. session.openLogicalChannel (AID_nonexisting);	CMD 1-1: APDU_MANAGE_CH_OPEN CMD 1-2: APDU_SELECT_BY_DF – CLA contains the Channel Number returned by the card in RESP 1;; Data = 'AID_nonexisting ' CMD 1-3: MANAGE CHANNEL (P1='80')	RESP 1-1: R-APDU - Data: Channel Number; SW '90 00' RESP 1-2: R-APDU – SW '6A 82' RESP 1-3: R-APDU - SW '90 00'	1. NoSuchElementError is expected.	CRC2
10	**A logical channel is already open to the non-multiselectable SE Applet**				
	1. session.openLogicalChannel (AID_TestApp); **2. session.openLogicalChannel (AID_TestApp);**	CMD 1-1: APDU_MANAGE_CH_OPEN CMD 1-2: APDU_SELECT_BY_DF – CLA contains the Channel Number returned by the card in RESP 1-1;; Data = 'AID_TestApp' CMD 2-1: APDU_MANAGE_CH_OPEN CMD 2-2: APDU_SELECT_BY_DF – CLA contains the Channel Number returned by the card in RESP 2-1; Data = 'AID_TestApp	RESP 1-1: R-APDU - Data: Channel Number; SW '90 00' RESP 1-2: R-APDU - SW '90 00' RESP 2-1: R-APDU - Data: Channel Number; SW '90 00' RESP 2-2: R-APDU - SW '6A 82' or 69 99 or '69 85'	1. No exception is expected. 2. NoSuchElementError is expected.	CRC2

		CMD 2-3 MANAGE CHANNEL (P1='80')	RESP 2-3: R-APDU - SW '90 00'		
11	**Open a logical channel, when session is already closed**				
	1. session.close(); **2. session.openLogicalChannel (AID_TestApp);**	none No APDU	none None	1. No exception is expected. 2. IllegalStateError is expected.	CRC3
12	**The application opening the logical channel has no access to the selected SE applet**				
	1. session.openLogicalChannel (AID_accessdenied);	no selection <for AID_accessdenied>	None	SecurityError is expected.	CRC4
13	**Application not selectable (SW=6999)**				
	1. session.openLogicalChannel (AID_TestApp_SW6999);	CMD 1-1: APDU_MANAGE_CH_OPEN CMD 1-2: APDU_SELECT_BY_DF – CLA contains the Channel Number returned by the card in RESP 1-1; Data = 'AID_TestApp_SW6999' CMD 1-3 MANAGE CHANNEL (P1='80')	RESP 1-1: R-APDU - Data: Channel Number; SW '90 00' RESP 1-2: R-APDU - SW '69 99' RESP 1-3: R-APDU - SW '90 00'	1. NoSuchElementError is expected.	CRC2
14	**Application selection returns a warning code 6283 (specified in ISO7816-4) – channel shall be opened**				
	1. Session.openLogicalChannet(AID_TestAppl_SW6283_selectresponse) **2. channel.transmit(Test_APDU1);**	CMD 1-1: APDU_MANAGE_CH_OPEN CMD 1-2: APDU_SELECT_BY_DF – CLA contains the Channel Number returned by the card in RESP 1-1; Data = 'AID_TestApp_SW6283_select response ' CMD 2: C-APDU ('XX 10 01 00 04 01 02 03 04 00')	RESP 1-1: R-APDU - Data: Channel Number; SW '90 00' RESP 1-2: R-APDU - 'DE AD C0 DE 62 83' RESP 2: R-APDU - Data = '01 02 03 04'; SW '90 00'	1. Returned Channel object is not null. No exception is expected. 2. Returned Response equals to 'R-APDU' - Data = '01 02 03 04'; SW '90 00'. No exception is expected.	CRN6
15	**Application selection returns a warning code 6280 (not specified in ISO 7816-4) – channel shall be opened**				
	1. **Session.openLogicalChannet(AID_TestApp_SW6280_selectresponse)** **2. channel.transmit(Test_APDU1);)**	CMD 1-1: APDU_MANAGE_CH_OPEN CMD 1-2: APDU_SELECT_BY_DF – CLA contains the Channel Number returned by the card in RESP 1-1; Data = 'AID_TestApp_SW6280_select response ' CMD 2: C-APDU ('XX 10 01 00 04 01 02 03 04 00')	RESP 1-1: R-APDU - Data: Channel Number; SW '90 00' RESP 1-2: R-APDU - 'DE AD C0 DE 62 80' RESP 2: R-APDU - Data = '01 02 03 04'; SW '90 00'	1. Returned Channel object is not null. No exception is expected. 2. Returned Response equals to 'R-APDU' - Data = '01 02 03 04'; SW	CRN6

				'90 00'. No exception is expected.	
16	**Application selection returns a warning code 6310 (not specified in ISO 7816-4) – channel shall be opened**				
	1. Session.openLogicalChannet(AID_TestApp_SW6310_selectresponse) **2. channel.transmit(Test_APDU1);)**	CMD 1-1: APDU_MANAGE_CH_OPEN CMD 1-2: APDU_SELECT_BY_DF – CLA contains the Channel Number returned by the card in RESP 1-; Data = 'AID_TestApp_SW6310_select response ' CMD 2: C-APDU ('XX 10 01 00 04 01 02 03 04 00')	RESP 1-1: R-APDU - Data: Channel Number; SW '90 00' RESP 1-2: R-APDU - 'DE AD C0 DE 63 10' RESP 2: R-APDU - Data = '01 02 03 04'; SW '90 00'	1. Returned Channel object is not null. No exception is expected. 2. Returned Response equals to 'R-APDU' - Data = '01 02 03 04'; SW '90 00'. No exception is expected.	CRN6
17	**Application selection returns a warning code 63C1 (specified in ISO7816-4) – channel shall be opened**				
	1. Session.openLogicalChannet(AID_TestApp_SW63C1_selectresponse) **2. channel.transmit(Test_APDU1);**	CMD 1-1: APDU_MANAGE_CH_OPEN CMD 1-2: APDU_SELECT_BY_DF – CLA contains the Channel Number returned by the card in RESP 1-1; Data = 'AID_TestApp_SW63C1_selectresponse ' CMD 2: C-APDU ('XX 10 01 00 04 01 02 03 04 00')	RESP 1-1: R-APDU - Data: Channel Number; SW '90 00' RESP 1-2: R-APDU - 'DE AD C0 DE 63 C1' RESP 2: R-APDU - Data = '01 02 03 04'; SW '90 00'	1. Returned Channel object is not null. No exception is expected. 2. Returned Response equals to 'R-APDU' - Data = '01 02 03 04'; SW '90 00'. No exception is expected.	CRN6

6.4.8 Method: Channel openLogicalChannel(byte[] aid) – Extended logical channels

(a) Conformance Requirements

The method with the following header shall be compliant to its definition in the API and according to chapter 8 in the API specification.

```
Channel openLogicalChannel(byte[] aid)
```

Normal execution

CRN1: The Transport API shall support logical channels according to ISO with up to 20 channels including the basic channel.

Parameter errors

Context errors

(b) Initial Conditions

SEService Object has been created and the isConnected() method has been called and has returned true.
A Reader instance "reader" is selected and a Session instance "session" is opened with the selected "reader".
Test case ID1 - 3: Three (3) logical channels are already opened to AID_TestApp_multiselectable.

(c) Test Procedure

Test case					
ID	**API Description**	**ISO Command Expectation DUT -> UICC Simulator / SE**	**ISO Response UICC Simulator / SE -> DUT**	**API Expectation**	**CRR**
1	**Open 19 logical channels, device manages maximum number of logical channels**				
	1. Loop 16 times: session.openLogicalChannel (AID_TestApp_multiselectable); end loop (store the references to the returned channel objects)	CMD 1-1-1: APDU_MANAGE_CH_OPEN CMD 1-2: APDU_SELECT_BY_DF – CLA contains the Channel Number (channel number > 3) returned by the card in RESP 1-1-1;; Data = 'AID_TestApp multiselectable' … CMD 1-1-16: APDU_MANAGE_CH_OPEN CMD 1-2-16: APDU_SELECT_BY_DF – CLA contains the Channel Number (channel number > 3) returned by the card in RESP 1-1-16;; Data = 'AID_TestApp multiselectable'	RESP 1-1-1: R-APDU - Data: Channel Number; SW '90 00' RESP 1-2-1: R-APDU - SW '90 00' ... RESP 1-1-16: R-APDU - Data: Channel Number; SW '90 00' RESP 1-2-16: R-APDU - SW '90 00'	1. Returned Channel object is not null. No exception is expected.	CRN1,
	2. Loop 16 time: channelXX.transmit(Test_APDU1) end loop (used the stored channel objects from step 1)	CMD 2-1: C-APDU ('4x 10 01 00 04 01 02 03 04 00') ... CMD 2-16: C-APDU ('4x 10 01 00 04 01 02 03 04 00')	RESP 2-1: R-APDU - Data = '01 02 03 04'; SW '90 00' .. RESP 2-16: R-APDU - Data = '01 02 03 04'; SW '90 00'	2. Returned Response equals to 'R-APDU' - Data = '01 02 03 04'; SW '90 00'. No exception is expected.	
	3. session.openLogicalChannel (AID_TestApp_multiselectable);	No APDU	none	3. Returned Channel object is null. No exception is expected.	
2	**Open 19 logical channels, device does not manage maximum number of logical channels and access control is checked on openSession method**				
	1. Loop 16 times: session.openLogica	CMD 1-1-1: APDU_MANAGE_CH_OPEN	RESP 1-1-1: R-APDU - Data: Channel Number; SW '90 00'	1. Returned Channel object is not null.	CRN1,

	lChannel (AID_TestApp_multiselectable); **end loop** **(store the references to the returned channel objects)**	CMD 1-2: APDU_SELECT_BY_DF – CLA contains the Channel Number (channel number > 3) returned by the card in RESP 1-1-1;; Data = 'AID_TestApp multiselectable' … CMD 1-1-16: APDU_MANAGE_CH_OPEN CMD 1-2-16: APDU_SELECT_BY_DF – CLA contains the Channel Number (channel number > 3) returned by the card in RESP 1-1-16;; Data = 'AID_TestApp multiselectable'	RESP 1-2-1: R-APDU - SW '90 00' ... RESP 1-1-16: R-APDU - Data: Channel Number; SW '90 00' RESP 1-2-16: R-APDU - SW '90 00'	No exception is expected.	
	2. **Loop 16 time: channelXX.transmit(Test_APDU1)** **end loop** **(used the stored channel objects from step 1)**	CMD 2-1: C-APDU ('4x 10 01 00 04 01 02 03 04 00') ... CMD 2-16: C-APDU ('4x 10 01 00 04 01 02 03 04 00')	RESP 2-1: R-APDU - Data = '01 02 03 04'; SW '90 00' .. RESP 2-16: R-APDU - Data = '01 02 03 04'; SW '90 00'	2. Returned Response equals to 'R-APDU' - Data = '01 02 03 04'; SW '90 00'. No exception is expected.	
	3. **session.openLogicalChannel (AID_TestApp_multiselectable);**	CMD 3-1: APDU_MANAGE_CH_OPEN	RESP 3-1: R-APDU – SW '68 81' or '6A 81'	3. Returned Channel object is null. No exception is expected.	
3	**Open 19 logical channels, device does not manage maximum number of logical channels and access control is checked on openLogicalChannel method**				
	1. **Loop 16 times: session.openLogicalChannel (AID_TestApp_multiselectable);** **end loop** **(store the references to the returned channel objects)**	CMD 1-1-1: APDU_MANAGE_CH_OPEN CMD 1-2: APDU_SELECT_BY_DF – CLA contains the Channel Number (channel number > 3) returned by the card in RESP 1-1-1;; Data = 'AID_TestApp multiselectable' … CMD 1-1-16: APDU_MANAGE_CH_OPEN CMD 1-2-16: APDU_SELECT_BY_DF – CLA contains the Channel Number (channel number > 3) returned by the card in RESP 1-1-16;; Data = 'AID_TestApp multiselectable'	RESP 1-1-1: R-APDU - Data: Channel Number; SW '90 00' RESP 1-2-1: R-APDU - SW '90 00' ... RESP 1-1-16: R-APDU - Data: Channel Number; SW '90 00' RESP 1-2-16: R-APDU - SW '90 00'	1. Returned Channel object is not null. No exception is expected.	CRN1,

	2. **Loop 16 time:** **channelXX.transmit(** **Test_APDU1)** **end loop** **(used the stored channel objects from step 1)**	CMD 2-1: C-APDU (’4x 10 01 00 04 01 02 03 04 00’) ... CMD 2-16: C-APDU (’4x 10 01 00 04 01 02 03 04 00’)	RESP 2-1: R-APDU - Data = ’01 02 03 04’; SW ‘90 00’ .. RESP 2-16: R-APDU - Data = ’01 02 03 04’; SW ‘90 00’	2. Returned Response equals to ‘R-APDU’ - Data = ‘01 02 03 04’; SW ‘90 00’. No exception is expected.	
	3. **session.openLogicalChannel (AID_TestApp_multiselectable);**	no selection <for AID_TestApp_multiselectable>	None	SecurityError is expected	

6.5 Class: Channel

Instances of this class represent an ISO7816-4 channel opened to a Secure Element. It can be either a logical channel or the default channel.
They can be used to send APDUs to the Secure Element. The "channel" instances are opened by calling the Session.openBasicChannel(byte[]) or Session.openLogicalChannel(byte[]) methods.

6.5.1 Method: void close()

(a) Conformance Requirements

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

```
void close()
```

Normal execution
CRN1: The close() method closes the channel to the Secure Element.
CRN2: If the channel is the basic channel, then it becomes available again.
CRN3: If the channel is already closed, the method is ignored.
CRN4: The close() method shall wait for completion of any pending transmit(byte[] command) before closing the channel.

Parameter errors
None

Context errors
None

(b) Initial Conditions

SEService Object has been created and the isConnected() method has been called and has returned true.
A Reader instance "reader" is selected and a Session instance "session" is opened with the selected “reader”.
Test cases ID1, ID3, ID4: A logical channel with “AID_TestApp” is already open.
Test case ID2: A basic channel with “AID_TestApp” is already open.

(c) Test Procedure

Test case					
ID	**API Description**	**ISO Command Expectation DUT → UICC Simulator/SE**	**ISO Response UICC Simulator/SE → DUT**	**API Expectation**	**CRR**
1	**Close an open logical channel**				
	1. Channel.close();	CMD 1-1: MANAGE CHANNEL (P1=’80’)	RESP 1-1: R-APDU - SW ‘90 00’	1. No exception is expected.	CRN1
2	**Close an open basic channel**				
	1. Channel.close();	CMD 1-1 No APDU	RESP 1-1 None	1. No exception is expected.	CRN2
3	**Close an already closed channel**				
	1. Channel.close(); **2. Channel.close();**	CMD-1-1: MANAGE CHANNEL (P1=’80’) CMD 2-1: No APDU	RESP 1-1: R-APDU - SW ‘90 00’ RESP 2-1: None	1. No exception is expected. 2. No exception is expected.	CRN3

4	**'Close' method shall wait for an ongoing 'transmit()'**				
	1. *Thread1: Transmit Test_APDU2* **Channel.transmit(Test_APDU2)** *Thread2 sleep/wait for 1 seconds* *2.Thread2:* **Channel.close();**	CMD 1-1: C-APDU (XX 10 01 00 04 01 02 03 04 00) CMD 2-1: MANAGE CHANNEL (P1='80')	RESP 1-1: R-APDU '01 02 03 04 '- SW '90 00' RESP 2-1: R-APDU - SW '90 00'	1. byte[]= {01, 02, 03, 04, 90,00} 2. close returns after transmit has been completed No exception is expected.	CRN4

6.5.2 Method: boolean isBasicChannel()

(a) Conformance Requirements

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

```
boolean isBasicChannel()
```

Normal execution
CRN1: This method returns true if the channel is the basic channel.
CRN2: This method returns false if the channel is a logical channel.

Parameter errors
None

Context errors
None

(b) Initial Conditions

SEService Object has been created and the isConnected() method has been called and has returned true.
A Reader instance "reader" is selected and a Session instance "session" is opened with the selected "reader".
Test case ID1: A basic channel with "AID_TestApp" is already open.
Test case ID2: A logical channel with "AID_TestApp" is already open.

(c) Test Procedure

Test case					
ID	**API Description**	**ISO Command Expectation DUT → UICC Simulator/SE**	**UICC Simulator - ISO Response UICC Simulator/SE → DUT**	**API Expectation**	**CRR**

1	**Check for an open basic channel**				
	1. Channel.isBasicChannel();	CMD1-1: None	RESP 1-1: None	1. Return 'true'.	CRN1
2	**Check for an open logical channel**				
	1. Channel.isBasicChannel();	CMD 1-1: None	RESP 1-1: None	1. Return 'false'	CRN2

6.5.3 Method: boolean isClosed()

(a) Conformance Requirements

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

```
boolean isClosed ()
```

Normal execution
CRN1: This method returns true if the channel is closed.
CRN2: This method returns false if the channel is open.

Parameter errors
None

Context errors
None

(b) Initial Conditions

SEService Object has been created and the isConnected() method has been called and has returned true.
A Reader instance "reader" is selected and a Session instance "session" is opened with the selected "reader".
For all test cases: a logical channel with "AID_TestApp" is already open.

(c) Test Procedure

Test case					
ID	**API Description**	**ISO Command Expectation DUT → UICC Simulator/SE**	**UICC Simulator - ISO Response UICC Simulator/SE → DUT**	**API Expectation**	**CRR**
1	**Check for an open channel**				
	1. Channel.isClosed();	CMD 1-1: None	CMD 1-1: None	1. Return 'false'	CRN2
2	**Check for a closed channel**				
	1. Channel.close(); **2. Channel.isClosed();**	CMD 1-1: MANAGE CHANNEL (P1='80') CMD 2-1: None	RESP 1-1: R-APDU - SW '90 00' RESP 2-1: None	1. No exception is expected 2. Return 'true'	CRN1

6.5.4 Method: byte[] getSelectResponse()

(a) Conformance Requirements

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

```
byte[] getSelectResponse()
```

Normal execution
CRN1: Returns the data as received from the application select command inclusively the status word.
CRN2: The returned byte array contains the data bytes in the following order:
[<first data byte>, …, <last data byte>, <sw1>, <sw2>]
CRN3: The returned byte array contains only the status word if the application select command has no data returned.
CRN4: Null is returned if the application select command has not been performed.
CRN5: Null is returned if the selection response cannot be retrieved by the reader implementation.

Parameter errors
None

Context errors
None

(b) Initial Conditions

SEService Object has been created and the isConnected() method has been called and has returned true.
A Reader instance “reader” is selected and a Session instance "session" is opened with the selected “reader”.
Test case ID1: A logical channel with “AID_TestApp_selectresponse” is already open.
Test cases ID2, ID6: A logical channel with “AID_TestApp” is already open.
Test case ID3: A logical channel with “null” AID is already open.
Test case ID4: A logical channel with “AID_TestApp_SW6283_selectresponse” is already open.
Test case ID5: A logical channel with “AID_TestApp_SW6280_selectresponse” is already open.
Test case ID7: A logical channel with “AID_TestApp_SW6310_selectresponse” is already open.
Test case ID8: A logical channel with “AID_TestApp_SW63C1_selectresponse” is already open.

(c) Test Procedure

Test case					
ID	**API Description**	**ISO Command Expectation DUT → UICC Simulator/SE**	**UICC Simulator - ISO Response UICC Simulator/SE → DUT**	**API Expectation**	**CRR**
1	**Return data and Status Word from an application select command**				
	1. Channel.getSelectResponse()	CMD 1-1: None	RESP 1-1: None	1. byte[ ]= { DE, AD, C0, DE, 90,00}	CRN1, CRN2
2	**Return only the Status Word from an application select command (if the select command has no returned data)**				
	1. Channel.getSelectResponse()	CMD 1-1: None	RESP 1-1: None	1. byte[ ]= {90,00}	CRN1, CRN3

3	**Return null in case the application select command is not performed**				
	1. Channel.getSelectResponse()	None	None	1. Return 'null'	CRN1, CRN4
4	**Check the handset correctly handles the select application command when the status word is 6283 (file invalidated)**				
	1.Channel.getSelectResponse();	None	None	1. byte[ ]= { DE, AD, C0, DE, 62,83}	CRN1, CRN2
5	**Check the handset correctly handles the select application command when the status word is 6280 (warning code not specified in ISO 7816-4)**				
	1.Channel.getSelectResponse();	None	None	1. byte[ ]= { DE, AD, C0, DE, 62,80}	CRN2
6	**Return null in case the selection response is not supported by the reader implementation**				
	1. Channel.getSelectResponse()	None	None	1. Return 'null'	, CRN5
7	**Check the handset correctly handles the select application command when the status word is 6310**				
	1.Channel.getSelectResponse();	None	None	1. byte[ ]= { DE, AD, C0, DE, 63, 10}	CRN1, CRN2
8	**Check the handset correctly handles the select application command when the status word is 63C1**				
	1.Channel.getSelectResponse();	None	None	1. byte[ ]= { DE, AD, C0, DE, 63, C1}	CRN2

6.5.5 Method: Session getSession()

(a) Conformance Requirements

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

```
Session getSession()
```

Normal execution
CRN1: This method returns the session object this channel is bound to.

Parameter errors
None

Context errors
None

(b) Initial Conditions

SEService Object has been created and the isConnected() method has been called and has returned true.
A Reader instance “reader” is selected and a Session instance "session" is opened with the selected “reader”.
A logical channel with “AID_TestApp” is already open.

(c) Test Procedure

Test case					
ID	**API Description**	**ISO Command Expectation DUT → UICC Simulator/SE**	**UICC Simulator - ISO Response UICC Simulator/SE → DUT**	**API Expectation**	**CRR**
1	**Return the Session object for a Channel instance**				
	1. Session == Channel.getSession ()	CMD 1-1: None	RESP 1-1: None	1. The Session object returned by getSession() is not null and is the same object created in initial conditions.	CRN1,

6.5.6 Method: byte[] transmit(byte[] command)

(a) Conformance Requirements

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

```
byte[] transmit(byte[] command)
```

Normal execution
CRN1: Transmit an APDU command as per ISO7816-4 to the Secure Element. The underlying layers can generate as many TPDUs as necessary to transport this APDU. The transport part is invisible from the application.
CRN2: The system ensures the synchronization between all the concurrent calls to this method. The entire APDU communication to this SE is locked to the APDU.
CRN3: The system ensures that only one APDU will be sent at a time, irrespective of the number of TPDUs that might be required to transport it to the SE.
CRN4: The channel information in the class byte in the APDU will be ignored: the system will add any required information to ensure the APDU is transported on this channel.
CRN5: Waiting time extension is handled in correct way.
CRN6: T=0/T=1 transport protocol related responses are handled inside the API or underlying implementation. The mapping from C-APDUs to C-TPDUs is described in the ISO 7816-3, section 12 with short Lc/Le field.

Parameter errors
CRP1: IllegalParameterError or NullPointerError – if the command parameter is null.
CRP2: SecurityError – if a MANAGE_CHANNEL command is supplied as a command parameter.
CRP3: SecurityError – if a SELECT by DF Name (p1=04) command is supplied as a command parameter.
CRP4: IllegalParameterError – if the command parameter is less than 4 bytes long.

Context errors

CRC1: IOError - if there is a communication problem to the reader or the Secure Element.
CRC2: IllegalStateError - if the channel is closed at the time of invocation of this method.
CRC3: SecurityError - if the command parameter is filtered by the security policy.

(b) Initial Conditions

SEService Object has been created and the isConnected() method has been called and has returned true.
A Reader instance “reader” is selected and a Session instance "session" is opened with the selected “reader”.
Test case ID1: A basic channel with “AID_TestApp” is already open.
Test cases ID2, ID5 to ID14 and ID16, ID19, ID20: A logical channel with “AID_TestApp” is already open.
Test cases ID3: A logical channel with “AID_TestApp” is already open. SE shall return logical channel number 1.
Test case ID4: Three channels are created in three different sessions.
Test case ID15: The two channels are created in two different sessions, each one created in a different SEService. (e.g. channel1 created by session1 created by seService1 created in thread1).
Test case ID17: A logical channel with “AID_TestApp_p1p2” is already open.
Test case ID18: A logical channel with “AID_TestApp_clains” is already open.
Test case ID13: UICC Simulator/UICC must only support T=0, a logical channel with “AID_TestApp” is already open.
Test case ID14: UICC Simulator/UICC must only support T=1, a logical channel with “AID_TestApp” is already open.
Test case ID21: A logical channel with “AID_TestApp_SW61xx” is already open.
Test case ID22: A logical channel with “AID_TestApp_Multi_SW61xx” is already open.
Test case ID23: Alogical channel with “AID_TestApp_Get_Response” is already open.

(c) Test Procedure

In case of T=0 protocol the case 2 type APDUs sent to the SE/UICC Simulator with wrong length are resent with correct length. The test procedure description only contains the APDUs sent first (with wrong length) and does not contain the APDUs resent with correct length.
For test case ID18, the scope of this test case is to check that the API implementation is not blocking any CLA/INS pairs except those mentioned in the specification. However, as some CLA/INS pairs are invalid, SE or UICC Simulator may send different R-APDU depending on their internal implementation. This behaviour is normal but it is impossible to specify accurately the “API expectation”. As long as the API implementation returns the response of the Secure Element, whatever it is, the test shall be considered successful.
It means that the “API Expectation” is that the transmit method shall always return the R-APDU sent by the SE/UICC Simulator as a response to the C-APDU, whatever it is.

Test case					
ID	**API Description**	**ISO Command Expectation DUT → UICC Simulator/SE**	**UICC Simulator - ISO Response UICC Simulator/SE → DUT**	**API Expectation**	**CRR**
1	**Transmit an APDU on Basic Channel**				
	1. Channel.transmit(Test_APDU1);	CMD 1-1: C-APDU (’00 10 01 00 04 01 02 03 04 00’)	RESP 1-1: R-APDU – ’01 02 03 04’ SW ‘90 00’	1. byte[ ]= {’01, 02, 03, 04, 90,00}	CRN1
	2. Channel.transmit(Test_APDU4);	CMD 2-1: C-APDU (’00 30 00 00’)	RESP 2-1: R-APDU – SW ‘90 00’	2. byte[ ]= {’ 90,00}	

	3. Channel.transmit(Test_APDU5);	CMD 3-1: C-APDU (’00 40 00 00 00’)	RESP 3-1: R-APDU – ’01 02 03 04’ SW ‘90 00’	3. byte[ ]= {’01, 02, 03, 04, 90,00}	
	4. Channel.transmit(Test_APDU6);	CMD 4-1: C-APDU (’00 50 00 00 04 01 02 03 04’)	RESP 4-1: R-APDU SW ‘90 00’	4. byte[ ]= {’ 90,00}	
2	**Transmit an APDU on Logical Channel**				
	1. Channel.transmit(Test_APDU1); **2. Channel.transmit(Test_APDU4);** **3. Channel.transmit(Test_APDU5);** **4. Channel.transmit(Test_APDU6);**	CMD 1-1: C-APDU (’XX 10 01 00 04 01 02 03 04 00’) CMD 2-1: C-APDU (’XX 30 00 00’) CMD 3-1: C-APDU (’XX 40 00 00 00’) CMD 4-1: C-APDU (’XX 50 00 00 04 01 02 03 04’)	RESP 1-1: R-APDU – ’01 02 03 04’ SW ‘90 00’ RESP 2-1: R-APDU – SW ‘90 00’ RESP 3-1: R-APDU – ’01 02 03 04’ SW ‘90 00’ RESP 4-1: R-APDU SW ‘90 00’	1. byte[ ]= {’01, 02, 03, 04, 90, 00} 2. byte[ ]= {90,00} 3. byte[ ]= {01, 02, 03, 04, 90,00} 4. byte[ ]= {90,00}	CRN1
3	**Transmit an APDU with a wrong channel number**				
	Send an Test_APDU1 with different channel number. E.g. with number channel = 2 → CLA = ‘02’: 1.**Channel.transmit(‘02100100040102030 4 00’);**	CMD 1-1: C-APDU (’XX 10 01 00 04 01 02 03 04 00’) (The logical channel number is corrected by the API)	RESP 1-1: R-APDU - ’01 02 03 04’ SW ‘90 00’	1. byte[ ]= {’01, 02, 03, 04, 90,00}	CRN1, CRN4
4	**Synchronization between concurrent calls**				

	1. Channel1 = Session1.openLogicalChannel (AID_TestApp_multiselectable);	CMD 1-1: APDU_MANAGE_CH_OPEN CMD 1-2: APDU_SELECT_BY_DF – CLA with Channel Number =1 (returned by the card in RESP 2-1); Data = 'AID_TestApp_**multiselectable** '	RESP 1-1: R-APDU - Data: Channel Number=1; SW '90 00' RESP 1-2: R-APDU - SW '90 00'	1. Returned Channel1 object is not null. No exception is expected.	CRN2, CRN3
	start new Thread2: **2. Channel2 = Session2.openLogicalChannel (AID_TestApp_multiselectable);**	CMD 2-1: APDU_MANAGE_CH_OPEN CMD 2-2: APDU_SELECT_BY_DF – CLA with Channel Number =2 (returned by the card in RESP 2-1); Data = 'AID_TestApp_**multiselectable** '	RESP 2-1: R-APDU - Data: Channel Number=2; SW '90 00' RESP 2-2: R-APDU - SW '90 00'	2. Returned Channel2 objects is not null. No exception is expected.	
	start new Thread3: **3. Channel3 = Session3.openLogicalChannel (AID_TestApp_multiselectable);**	CMD 3-1: APDU_MANAGE_CH_OPEN CMD 3-2: APDU_SELECT_BY_DF – CLA with Channel Number =3 (returned by the card in RESP 2-1); Data = 'AID_TestApp_**multiselectable** '	RESP 3-1: R-APDU - Data: Channel Number=3; SW '90 00' RESP 3-2: R-APDU - SW '90 00'	3. Returned Channel3 object is not null. No exception is expected.	
	Thread 1: **4. Channel1.transmit(Test_APDU2);** *Thread2: wait – 0,5 s* *Thread 3: wait – 0,7 s*	CMD 4-1: C-APDU ('01 10 02 00 04 01 02 03 04 00')	RESP 4-1: Wait 1 second and send response: R-APDU – '01 02 03 04' SW '90 00'	4. byte[ ]= {01, 02, 03, 04, 90,00}	
	5. Channel2.transmit(Test_APDU2);	CMD 5-1: C-APDU ('02 10 02 00 04 05 06 07 08 00')	RESP 5-1: Wait 1 second and send response: R-APDU – '05 06 07 08' SW '90 00'	5. byte[ ]= {05, 06, 07, 08, 90,00}	
	6. Channel3.transmit(Test_APDU2);	CMD 6-1: C-APDU ('03 10 02 00 04 09 0A 0B 0C 00')	RESP 6-1: Wait 1 second and send response: R-APDU – '09 0A 0B 0C' SW '90 00'	6byte[ ]= {09 0A 0B, 0C, 90,00}	
5	**Null parameter command**				
	1.	CMD 1-1: No APDU	RESP 1-1: None	`1.IllegalParame`	CRP1

	Channel.transmit(null);			terError or NullPointerError	
6	**MANAGE CHANNEL_OPEN as parameter command**				
	1. Channel.transmit(APDU_MANAGE_CH_OPEN);	CMD 1-1: No APDU	RESP 1-1: None	1.SecurityError	CRP2
7	**SELECT BY DF NAME as parameter command**				
	1. Channel.transmit(APDU_SELECT_BY_DF(AID_TestApp));	CMD 1-1: No APDU	RESP 1-1: None	1. SecurityError	CRP3
8	**Communication problem with the Secure Element**				
	1. **Channel.transmit(Test_APDU1);**	CMD 1-1: : C-APDU ('XX 10 02 00 04 01 02 03 04 00')	RESP 1-1: None	1. IOError (APDU must not be resent automatically even if the SE/UICC Simulator answers after reset.)	CRC1
9	**Transmit an APDU when the channel is closed**				
	1. Channel.close();	CMD 1-1: MANAGE CHANNEL (P1='80')	RESP 1-1: R-APDU - SW '90 00'	1. No exception is expected.	CRC2
	2.Channel.transmit(Test_APDU1);	CMD 2-1: No APDU	RESP 2-1: None	2. IllegalStateError	
10	**Command parameter shorter than 4 bytes**				
	Transmit a dummy command to the application with only 3 bytes: ***1.*** **Channel.transmit('001500');**	CMD 1-1: No APDU	RESP 1-1 None	1. IllegalParameterError	CRP4
	Transmit a empty command **2. Channel.transmit('');**	CMD 2-1: No APDU	RESP 2-1 None	2. IllegalParameterError	
11	**Access Control rule does not allow the sending of this APDU**				
	1. Channel.transmit(Test_APDU3);	CMD 1-1: No APDU	RESP 1-1: None	1. SecurityError is expected.	CRC3
12	**Check waiting time extension**				
	1. Channel.transmit(Test_APDU7);	CMD 1-1: C-APDU ('XX 55 00 00')	*- waiting time extension - received-* RESP 1-1: R-APDU –SW '90 00'	1. byte[ ]= { 90, 00}	CRN5
13	**Check protocol handling of T=0**				
	1.				CRN6

	Channel.transmit(Test_APDU1); **2. Channel.transmit(Test_APDU5);**	CMD 1-1: C-APDU ('XX 10 01 00 04 01 02 03 04 00') *- getResponse command received by card -* CMD 2-1: C-APDU ('XX 40 00 00 00') *- command resend with correct length-* TPDU ('XX 40 00 00 04')	*- procedure byte to trigger getResponse command* {61 04}- RESP 1-1: R-APDU – '01 02 03 04' SW '90 00' *- procedure byte '6C xx' to trigger to resend the command with correct length -* RESP 2-1: R-APDU – '01 02 03 04' SW '90 00'	1. byte[ ]= {'01, 02, 03, 04, 90, 00} 2. byte[ ]= {01, 02, 03, 04, 90,00}	
14	**Check Protocol handling of T=1**				
	1. Channel.transmit(Test_APDU1); **2. Channel.transmit(Test_APDU5);**	CMD 1-1: C-APDU ('XX 10 01 00 04 01 02 03 04 00') CMD 2-1: C-APDU ('XX 40 00 00 00')	RESP 1-1: R-APDU – '01 02 03 04' SW '90 00' RESP 2-1: R-APDU – '01 02 03 04' SW '90 00'	1. byte[ ]= {'01, 02, 03, 04, 90, 00} 2. byte[ ]= {01, 02, 03, 04, 90,00}	CRN6
15	**Synchronization between concurrent SEServices**				
	1. Channel1 = Session1.openLogicalChannel (AID_TestApp_multiselectable); *start new Thread2:* **2. Channel2 = Session2.openLogicalChannel (AID_TestApp_multiselectable);** *Thread 1:* **3. Channel1.transmit(Test_APDU2);** *Thread2: wait – 0,5 s* **4. Channel2.transmit(Test_APDU2);**	CMD 1-1: APDU_MANAGE_CH_OPEN CMD 1-2: APDU_SELECT_BY_DF – CLA with Channel Number =1 (returned by the card in RESP 2-1); Data = 'AID_TestApp_**multiselectable** ' CMD 2-1: APDU_MANAGE_CH_OPEN CMD 2-2: APDU_SELECT_BY_DF – CLA with Channel Number =2 (returned by the card in RESP 2-1); Data = 'AID_TestApp_**multiselectable** ' Both APDUs are transmitted: CMD 3-1: C-APDU ('01 10 02 00 04 01 02 03 04 00') CMD 4-1: C-APDU ('02 10 02 00 04 05 06 07 08 00')	RESP 1-1: R-APDU - Data: Channel Number=1; SW '90 00' RESP 1-2: R-APDU - SW '90 00' RESP 2-1: R-APDU - Data: Channel Number=2; SW '90 00' RESP 2-2: R-APDU - SW '90 00' RESP 3-1: R-APDU – '01 02 03 04' SW '90 00' RESP 4-1: R-APDU – '05 06 07 08' SW '90 00'	1. Returned Channel1 object is not null. No exception is expected. 2. Returned Channel2 objects is not null. No exception is expected. 3. byte[ ]= {01, 02, 03, 04, 90,00} 4. byte[ ]= {05, 06, 07, 08, 90,00}	CRN2, CRN3
16	**Transmit APDUs with various admitted length**				

	1. Channel.transmit(Test_APDU1); ***-this test shall run from lc=01 to lc=ff, so in fact transmit is called 255 times-***	CMD 1-1: C-APDU ('01 10 01 00 lc 01 02 03 04 .. lc 00')	RESP 1-1: R-APDU – '01 02 03 04 . lc.' SW '90 00'	1.. byte[ ]= {'01, 02, 03, 04, lc.., 90, 00}	CRN1
17	**Sending of APDUs with different P1 and recover Status Word returned by the UICC application (Expected Status words for each P1 are listed in Table** 9: P1 - Status Word Pairs **)**				
	1. From P1 = 0x01 to 0x32 loop: Channel.transmit(APDU_Case1);	CMD 1-1: C-APDU ('XX 01 01 00') …. CMD 1-50: C-APDU ('XX 01 32 00')	RESP 1-1: R-APDU – SW1-SW2 ... RESP 1-50: R-APDU – SW1-SW2	1. byte[ ]= {SW1, SW2} ... 50. byte[ ]= {SW1, SW2}	CRN1
	2. From P1 = 0x01 to 0x32 loop: Channel.transmit(APDU_Case2);	CMD 2-1: C-APDU ('XX 02 01 00 FF') …. CMD 2-17: C-APDU ('XX 02 11 00 FF') CMD 2-18: C-APDU ('XX 02 12 00') …. CMD 2-50: C-APDU ('XX 02 32 00')	RESP 2-1: R-APDU – <data field of 255 bytes> SW1-SW2 ... RESP 2-17: R-APDU – <data field of 255 bytes> SW1-SW2 RESP 2-18: R-APDU – SW1-SW2 RESP 2-50: R-APDU – SW1-SW2	1. byte[ ]= {data field of 255 bytes, SW1, SW2} ... 17. byte[ ]= {data field of 255 bytes, SW1, SW2} 18. byte[ ]= {SW1, SW2} 50. byte[ ]= {SW1, SW2}	
	3. From P1 = 0x01 to 0x32 loop: Channel.transmit(APDU_Case3);	CMD 3-1: C-APDU ('XX 03 01 00 FF' <Data field of 255 bytes>) …. CMD 3-50: C-APDU ('XX 03 32 00 FF' <Data field of 255 bytes>)	RESP 3-1: R-APDU – SW1-SW2 ... RESP 3-50: R-APDU – SW1-SW2	1. byte[ ]= {SW1, SW2} ... 50. byte[ ]= {SW1, SW2}	
	4. From P1 = 0x01 to 0x32 loop: Channel.transmit(APDU_Case4);	CMD 4-1: C-APDU ('XX 04 01 00 FF' <Data field of 255 bytes> FF) …. CMD 4-17: C-APDU ('XX 04 11 00 FF' <Data field of 255 bytes> FF) CMD 4-18: C-APDU ('XX 04 12 00 FF' <Data field of 255 bytes>) …. CMD 4-50: C-APDU ('XX 04 32 00 FF' <Data field of 255 bytes>)	RESP 4-1: R-APDU – <data field of 255 bytes> SW1-SW2 ... RESP 4-17: R-APDU – <data field of 255 bytes> SW1-SW2 RESP 4-18: R-APDU – SW1-SW2 ... RESP 4-50: R-APDU – SW1-SW2	1. byte[ ]= {data field of 255 bytes, SW1, SW2} ... 17. byte[ ]= {data field of 255 bytes, SW1, SW2} 18. byte[ ]= {SW1, SW2} 50. byte[ ]= {SW1, SW2}	

18	**Sending of all allowed class instruction pairs (according to ISO 7816-4) and recover Status Word returned by the UICC application**				
	Send APDUs with the Class/Instruction pairs from 0x0000 to 0xFEFF; Exclude SELECT BY DF (INS=A4 and P1= 04), MANAGE CHANNEL (INS=70), INS=0x6x and INS=0x9x on all CLA 1. From INS = 0x00 to INS= 0x5F: For CLA=0x00 to 0xFE loop: Channel.transmit(APDU); 2. From INS = 0x71 to INS= 0x8F: For CLA=0x00 to 0xFE loop: Channel.transmit(APDU); 3. From INS = 0xA0 to INS= 0xFF: For CLA=0x00 to 0xFE loop: Channel.transmit(APDU); Exclude SELECT BY DF (INS=A4 and P1= 04) from the loop.	CLA byte will be adapted by device depending on the assigned channel CMD 1-1: C-APDU ('XX 00 00 00 10 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F10 00') …. CMD 1-X: C-APDU ('XX 5F 00 00 10 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F10 00') CMD 2-1: C-APDU ('XX 71 00 00 10 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F10 00') …. CMD 2-X: C-APDU ('XX 8F 00 00 10 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F10 00') CMD 3-1: C-APDU ('XX A0 00 00 10 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F10 00') …. CMD 3-X: C-APDU ('XX FF 00 00 10 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F10 00')	RESP 1-1: R-APDU … RESP 1-X: R-APDU RESP 2-1: R-APDU … RESP 2-X: R-APDU RESP 3-1: R-APDU ... RESP 3-X: R-APDU	1.byte[]=RESP 1-1 ... X.byte[]=RESP 1-X 1.byte[]=RESP 2-1 … X.byte[]=RESP 2-X 1.byte[]=RESP 3-1 ... X.byte[]=RESP 3-X	CRN1,
19	**MANAGE CHANNEL CLOSE as parameter command**				
	1. Channel.transmit(APDU_MANAGE_CH_	CMD 1-1: No APDU	RESP 1-1: None	`1.SecurityError`	CRP2

	CLOSE);				
20	**SELECT BY FID as parameter command**				
	1. Channel.transmit(APDU_SELECT_BY_FID);	CMD 1-1: APDU_SELECT_BY_FID	RESP 1-1: R-APDU – {90 00}	1. byte[ ]= {90, 00}	CRN1
21	**Management of status code 61xx , APDU case 2**				
	1. Channel.transmit(Test_APDU8);	CMD 1-1: C-APDU (’XX 40 00 00 04’) *- getResponse command received by card -*	RESP 1-1: R-APDU – {61 04} RESP 1-2: R-APDU – ’01 02 03 04’ SW ‘90 00’	1. byte[ ]= {’01, 02, 03, 04, 90, 00}	CRN1
22	**Concatenated get response**				
	1. For P1 = 0x00 Channel.transmit(APDU_LONG_RESPONSE);	CMD 1-1: C-APDU (’XX 60 00 00 00’) *- getResponse command received by card –* *..* *repeat 8 times* *- getResponse command received by card –* *…* *- getResponse command received by card -*	RESP 1-1: R-APDU – {61 20} RESP 1-2: R-APDU – ’00 … 00’ (32 bytes) SW ‘61 20 --- repeat 8 times RESP 1-x: R-APDU – ’XX … XX’ (32 bytes) SW ‘61 20’ … RESP 1-11: R-APDU – ’99 … 99’ (32 bytes) SW ’90 00’	1. byte[ ]= { data field of 320 bytes: ’00 … 00’ (32 bytes), … ’XX … XX’ (32 bytes), … ’99 … 99’ (32 bytes), , 90, 00}	CRN1
23	**GET RESPONSE as parameter command**				
	1. Channel.transmit(Test_APDU8); **2. Channel.transmit(APDU_GET_RESPONSE);**	CMD 1-1: C-APDU (’XX 40 00 00 04’) CMD 2-1: C-APDU (’XX C0 00 00 04’)	RESP 1-1: R-APDU – SW ‘62 F1’ RESP 2-1: R-APDU – ’01 02 03 04’ SW ‘90 00’	1. byte[ ]= {62, F1} 2. byte[ ]= {01, 02, 03, 04, 90,00}	CRN1

6.5.7 Method: boolean[] selectNext()

(a) Conformance Requirements

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

```
boolean selectNext()
```

Normal execution

CRN1: Performs a selection of the next applet on this channel that matches to the partial AID specified in the openBasicChannel(byte[] aid) or openLogicalChannel(byte[] aid) method. This mechanism can be used by a device application to iterate through all applets matching to the same partial AID. If selectNext() returns true a new applet was successfully selected on this channel.
CRN2: The implementation of the underlying SELECT command within this method shall use the same values as the corresponding openBasicChannel(byte[] aid) or openLogicalChannel(byte[] aid) command with the option: P2='02' (Next occurrence).
CRN3: If no further applet exists with matches to the partial AID this method returns false and the already selected applet stays selected.
CRN4: The select response stored in the Channel object shall be updated with the APDU response of the SELECT command.

Context errors
CRC1: IOError - if there is a communication problem to the reader or the Secure Element.
CRC2: OperationNotSupportedError - if this operation is not supported by the card.
CRC3: IllegalStateError - if the Secure Element is used after being closed.

(b) Initial Conditions

SEService Object has been created and the isConnected() method has been called and has returned true.
A Reader instance "reader" is selected and a Session instance "session" is opened with the selected "reader".
Test cases ID1, ID3, ID5, ID6 ,ID7: A logical channel with "AID_Partial_1_instance_1" is already open by selecting "AID_Partial_1".
Test cases ID2, ID4: A logical channel with "AID_Partial_2" is already open.
Test cases ID8: A logical channel with "AID_Partial_SW6280" is already open.
Test cases ID9: A logical channel with "AID_Partial_SW6283" is already open.

(c) Test Procedure

Test case					
ID	**API Description**	**ISO Command Expectation DUT → UICC Simulator/SE**	**UICC Simulator - ISO Response UICC Simulator/SE → DUT**	**API Expectation**	**CRR**
1	**Next Applet matches with partial AID**				
	1. Channel.selectNext( );	CMD 1-1: APDU_SELECT_BY_DF – CLA with Channel Number =1 ; P2='02' (Next occurrence); Data = 'AID_Partial_1'	RESP 1-1: R-APDU - SW '90 00'	1. **Return 'true'**	CRN1, CRN2
2	**No other Applet does not match with partial AID**				
	1. Channel.selectNext( );	CMD 1-1: APDU_SELECT_BY_DF – CLA with Channel Number =1 ; P2='02' (Next occurrence) ; Data = 'AID_Partial_2'	RESP 1-1: R-APDU - SW '6A 82'	1. Return 'false'	CRN2, CRN3
3	**Check select response is updated**				

	1. response1 = Channel.getSelectResponse() **2. Channel.selectNext();** **3. response2 = Channel.getSelectResponse()**	CMD 1-1: None CMD 2-1: APDU_SELECT_BY_DF – CLA with Channel Number =1 ; P2='02' (Next occurrence); Data = 'AID_Partial_1' CMD 3-1: None	RESP 1-1: None RESP 2-1: R-APDU -AID SW '90 00' RESP 3-1: None	1. response1 = { AID_Partial_1_instance_1, 90, 00} 2. Return 'true' 3. response2 = { AID_Partial_1_instance_2, 90, 00}	CRN1, CRN2, CRN4
4	**Check select response is not updated in case selectNext() fails**				
	1. response1 = Channel.getSelectResponse() **2. Channel.selectNext();** **3. response2 = Channel.getSelectResponse()** *4.* **Channel.transmit(Test_APDU4);**	CMD 1-1: None CMD 2-1: APDU_SELECT_BY_DF – CLA with Channel Number =1 ; P2='02' (Next occurrence); Data = 'AID_Partial_2' CMD 3-1: None CMD 4-1: C-APDU ('XX 30 00 00')	RESP 1-1: None RESP 2-1: R-APDU - SW '6A 82' RESP 3-1: None RESP 4-1: R-APDU – SW '90 00'	1. response1 = { AID_Partial_2_instance_1 90. 00} 2. Return 'false' 3. response1 = { AID_Partial_2_instance_1 90. 00} (previous applet stays selected) 4. byte[ ]= {90,00}	CRN1, CRN2, CRN4
5	**Communication problem with the Secure Element**				
	1. Channel.selectNext();	CMD 1-1: APDU_SELECT_BY_DF – CLA with Channel Number =1 ; P2='02' (Next occurrence); Data = 'AID_Partial_1'	RESP 1-1: None	`1. IOError`	CRC1
6	**Operation not supported by the Secure Element**				
	1. Channel.selectNext();	CMD 1-1: APDU_SELECT_BY_DF – CLA with Channel Number =1 ; P2='02' (Next occurrence); Data = 'AID_Partial_1'	RESP 1-1: R-APDU - SW '6A 81'	1. `OperationNotSupportedError`	CRC2
7	**selectNext() when the channel is closed**				
	1. Channel.close();	CMD 1-1: MANAGE CHANNEL (P1='80')	RESP 1-1: R-APDU - SW '90 00'		CRC3

	2. Channel.selectNext( );	CMD 2-1: No APDU	RESP 2-1: None	2. IllegalStateError	
8	selectNext() when the next application selection returns a not specified warning 6280				
	1. response1 = Channel.getSelectResponse()	CMD 1-1: None	RESP 1-1: None	1. response1 = { AID_Partial_SW6280_instance_1, 6280}	CRN1, CRN2, CRN4
	2. Channel.selectNext( );	CMD 2-1: APDU_SELECT_BY_DF – CLA with Channel Number =1 ; P2='02' (Next occurrence); Data = 'AID_Partial_SW_6280'	RESP 2-1: R-APDU - SW '6280'	2. Return 'true'	
	3. response2 = Channel.getSelectResponse()	CMD 3-1: None	RESP 3-1: None	3.Response2 = { AID_Partial_SW6280_instance_2, 6280}	
9	selectNext() when the next application selection returns a specified warning 6283				
	1. response1 = Channel.getSelectResponse()	CMD 1-1: None	RESP 1-1: None	1. response1 = { AID_Partial_SW6283_instance_1, 6283}	CRN1, CRN2, CRN4
	2. Channel.selectNext( );	CMD 2-1: APDU_SELECT_BY_DF – CLA with Channel Number =1 ; P2='02' (Next occurrence); Data = 'AID_Partial_SW_6283'	RESP 2-1: R-APDU - SW '6283'	2. Return 'true'	
	3. response2 = Channel.getSelectResponse()	CMD 3-1: None	RESP 3-1: None	3. Response2 = { AID_Partial_SW6283_instance_2, 6283}	

7. History

Version	Date	Author	Comment
0.9	13.09.2013	SIMalliance	Initial Release 0.9
1.0	29.01.2014	SIMalliance	First release after public review; several test cases added or existing modified
1.1	21.07.2014	SIMalliance	Chapters 4.3, 6.4.8 added; new options in chapter 4.2; chapter 4.4 updated; several clarification on different test cases

Table 10: History

Annex A: (Normative): None Tested Requirements

The requirements that are not tested in the current version of the specification are listed in table A.1. The section index referenced in table A.1 is the index used in this specification.

Requirement	Class	Index	Method
CRC1: IOError - something went wrong with the communication to the Secure Element. (e.g. no SE connected or no more Session available)	Reader	6.3.4	Session openSession()
CRN1: This method closes all the sessions opened on this reader	Reader	6.3.5	void closeSessions()
CRN2: The Secure Element needs to be prepared (initialized) for communication (i.e. switched on)	Reader	6.3.4	Session openSession()
CRN2: If there is no reader, then the array of readers returned by getReaders() method has length 0	SEService	6.1.2	Reader[] getReaders()

Table A1

Annex B: Access Control Configuration Examples

Access Control Applet (ARA)

A simple ARA applet provides the access rules to the Enforcer application in the mobile. This will be provided on the SIMalliance website. According to these access rules, the Enforcer will decide whether to allow access to any applet instance or not (see GPSEAC specification).
The ARA-M Applet from this test spec may provide to the Enforcer either all the existing access rules (GET DATA all command) or only the specific rules of an applet (GET DATA specific command).

- **Rule 1 implementation**: The GET DATA (specific) for getting the rule related to the denied applet is:

CLA → 80
INS → CA
P1 P2 → FF 50 (GET DATA specific)
Lc → XX
REF-DO → E1 (tag) XX (length)
AID-REF-DO →4F (tag) XX (length)
XX XX XX (Denied Applet AID)
Hash-REF-DO → C1 (tag) 00 (length)
Le → 00

The response contains the access rule which does not allow any APDU to be sent to the denied applet from any mobile application:

```
Response AR-DO → FF 50 08
        AR-DO →  E3 (tag)  06 (length)
            APDU-AR-DO → D0 (tag)  01 (length)  00 (Never: APDU access is not allowed)
            NFC-AR-DO → D1(tag) 01 (length)  00 (Never: NFC event access is not allowed)
```

- **Rule 2 implementation**: GPSEAC forces the definition of access rules only for allowed APDUs per applet. Therefore it is required to allow all APDUs used for '**AID_TestApp**' Applet test cases, it means, all the APDUs listed in the OMAPI test spec with the exception of 'Test_APDU3' command:

Then, the set of masks and respective expected results for allowing Test_APDUx are as follows:

Incoming APDU	**Mask**	**Expected result**
Test_APDU1	*FE FF FF FF*	*00 10 01 00*
Test_APDU2 (CLA = 00, 01)	*FE FF FF FF*	*00 10 02 00*
Test_APDU2 (CLA = 02)	*FF FF FF FF*	*02 10 02 00*
Test_APDU4	*FE FF FF FF*	*00 30 00 00*
Test_APDU5 and Test_APDU6	*FE EF FF FF*	*00 40 00 00*
APDU_SELECT_BY_FID	*FE FF FB FF*	*00 A4 00 00*
APDU_MANAGE_CHANNEL	*FE FF 7F E0*	*00 70 00 00*

ARA Applet sends all masks and expected results when the Enforcer requests it through a GET DATA (specific) for '**AID_TestApp'** Applet.

```
CLA → 80
INS → CA
P1 P2 → FF 50  (GET DATA specific)
Lc → XX
REF-DO → E1 (tag) XX (length)
        AID-REF-DO →4F (tag) XX (length)
                        XX XX XX XX (AID_TestApp_apdu_filtered)
        Hash-REF-DO → C1 (tag) 00 (length)
Le → 00
```

The response of the ARA Applet encapsulates the masks and expected results:

```
Response AR-DO → FF 50 2F
        AR-DO →  E3(tag) 2D (length)
            APDU-AR-DO → D0 (tag) 28 (length)
                    00 10 01 00 FE FF FF FF 00 10 02 00 FE FF FF FF 02 10 02 00 FF FF FF FF
                    00 30 00 00 FE FF FF FF 00 40 00 00 FE EF FF FF
```

Access Control file system (ARF)

Additionally a PKCS#15 file structure is provided with the access rules. Here it is described following PKCS#15 examples in GPSEAC specification (see also PKCS#15 v1.1 spec):

PKCS#15 file system

```
MF (3F00)
|- EF DIR (2F00) --> shall reference PKCS-15
|
|- DF PKCS-15 (7F50)
   |
   |- ODF (5031) --> shall reference DODF
   |- DODF (5207) --> shall reference EF ACMain
   |- EF ACMain (4200) --> shall reference EF ACRules
   |- EF ACRules (4300) --> shall reference EF ACConditions files
   |- EF ACConditions1 (4310)
   |- EF ACConditions2 (4311)
   |- EF ACConditions3 (4312)

The following file identifiers are decided by the application issuer: PKCS-15,
DODF, ACMain, ACConditions,..

ODF:
A7 06 30 04 04 02 52 07

DODF:
A1 29 30 00 30 0F 0C 0D 47 50 20 53 45 20 41 63 63 20 43 74 6C A1 14 30 12 06 0A 2A
86 48 86 FC 6B 81 48 01 01 30 04 04 02 42 00

ACMain:
30 10 04 08 01 02 03 04 05 06 07 08 30 04 04 02 43 00

ACRules:
30 15 A0 0D 04 XX XX XX XX ..     30 04 04 02 43 10
30 15 A0 0D 04 XX XX XX XX ..     30 04 04 02 43 11
30 08 82 00                                        30 04 04 02 43 12

ACConditions1:
FF FF

ACConditions2:
30 53
04 00
A0 4F
A0 48
A1 46
04 08 00 10 01 00 FE FF FF FF
04 08 00 10 02 00 FE FF FF FF
04 08 02 10 02 00 FF FF FF FF
04 08 00 30 00 00 FE FF FF FF
04 08 00 40 00 00 FE EF FF FF
04 08 00 A4 00 00 FE FF FB FF
04 08 00 70 00 00 FE FF 7F E0
A1 03
80 01 00

ACConditions3:
30 00
```

Annex C: Error Mapping Table

Error Types	Java Exceptions
IOError	java.io.IOException
SecurityError	java.lang.SecurityException
NoSuchElementError	java.util.NoSuchElementException
IllegalStateError	java.lang.IllegalStateException
IllegalParameterError	java.lang.IllegalArgumentException
OperationNotSupportedError	java.lang.UnsupportedOperationException
NullPointerError	java.lang.NullPointerException

Table C1