


# Open Mobile API Specification - Transport API Test Plan

Published by  simalliance now Trusted Connectivity Alliance

November 2011

**Copyright © 2011 Trusted Connectivity Alliance Ltd.**

The information contained in this document may be used, disclosed and reproduced without the prior written authorization of Trusted Connectivity Alliance. Readers are advised that Trusted Connectivity Alliance reserves the right to amend and update this document without prior notice. Ownership of the OMAPI Specification has been transferred to GlobalPlatform. All future releases will be available on the GlobalPlatform website.

# Table of Contents

1. Terminology .....	5
1.1 Abbreviations and Notations.....	5
1.2 Terms.....	5
2. Overview .....	6
3. Class SEService .....	7
3.1 Method: Reader[] getReaders() .....	7
3.1.1 Conformance Requirements .....	7
3.2 Method: boolean isConnected ().....	7
3.2.1 Conformance Requirements .....	7
3.3 Method: void shutdown () .....	7
3.3.1 Conformance Requirements .....	7
3.4 Constructor: SEService(Context context, SEService.CallBack listener).....	8
3.4.1 Conformance Requirements .....	8
4. Class (or interface): SEService.CallBack .....	9
4.1 Method: void serviceConnected(SESERVICE service) .....	9
4.1.1 Conformance Requirements .....	9
5. Class: Reader .....	10
5.1 Method: String getName() .....	10
5.1.1 Conformance Requirements .....	10
5.2 Method SESERVICE getService().....	10
5.2.1 Conformance Requirements .....	10
5.3 Method: boolean isSecureElementPresent() .....	10
5.3.1 Conformance Requirements .....	10
5.4 Method: Session openSession().....	11
5.4.1 Conformance Requirements .....	11
5.5 Method: void closeSessions().....	11
5.5.1 Conformance Requirements .....	11
6. Class: Session .....	12
6.1 Method: Reader getReader().....	12
6.1.1 Conformance Requirements .....	12
6.2 Method: byte[] getATR() .....	12
6.2.1 Conformance Requirements .....	12

6.3 Method: void close() .....	12
6.3.1 Conformance Requirements .....	12
6.4 Method: boolean isClosed().....	13
6.4.1 Conformance Requirements .....	13
6.5 Method: void closeChannels() .....	13
6.5.1 Conformance Requirements .....	13
6.6 Method: Channel openBasicChannel(byte[] aid).....	13
6.6.1 Conformance Requirements .....	13
6.7 Method: Channel openLogicalChannel(byte[] aid) .....	14
6.7.1 Conformance Requirements .....	14
<b>7. Class: Channel.....</b>	<b>15</b>
7.1 Method: void close() .....	15
7.1.1 Conformance Requirements .....	15
7.2 Method: boolean isBasicChannel().....	15
7.2.1 Conformance Requirements .....	15
7.3 Method: boolean isClosed().....	15
7.3.1 Conformance Requirements .....	15
7.4 Method: byte[] getSelectResponse() .....	16
7.4.1 Conformance Requirements .....	16
7.5 Method: Session getSession().....	16
7.5.1 Conformance Requirements .....	16
7.6 Method: byte[] transmit(byte[] command) .....	16
7.6.1 Conformance Requirements .....	16
<b>8. History .....</b>	<b>18</b>

# Table of Tables

TABLE 1-1: ABBREVIATIONS AND NOTATIONS.....	5
TABLE 1-2: TERMS.....	5
TABLE 8-1: HISTORY .....	18

# 1. Terminology

The given terminology is used in this document.

## 1.1 Abbreviations and Notations

Table 1-1: Abbreviations and Notations

Abbreviation	Description
<b>SE</b>	Secure Element
<b>API</b>	Application Programming Interface
<b>ATR</b>	Answer to Reset (as per ISO/IEC 7816-4)
<b>APDU</b>	Application Protocol Data Unit (as per ISO/IEC 7816-4)
<b>ISO</b>	International Organisation for Standardisation
<b>ASSD</b>	Advanced Security SD cards (SD memory cards with an embedded security system) as specified by the SD Association.
<b>OS</b>	Operating system
<b>RIL</b>	Radio Interface Layer
<b>SFI</b>	Short File ID
<b>FID</b>	File ID
<b>FCP</b>	File Control Parameters
<b>MF</b>	Master File
<b>DF</b>	Dedicated File
<b>EF</b>	Elementary File
<b>OID</b>	Object Identifier
<b>DER</b>	Distinguished Encoding Rules of ASN.1
<b>ASN.1</b>	Abstract Syntax Notation One

## 1.2 Terms

Table 1-2: Terms

Term	Description
<b>Secure Element</b>	Smart Card Chip available in the mobile device. For example UICC/SIM, embedded Secure Element, Secure SD card, ...
<b>Applet</b>	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
<b>Application</b>	Device/Terminal/Mobile application: An application which is installed in the mobile device and runs within the mobile device
<b>Session</b>	An open connection between an application on the device (e.g. mobile phone) and a SE.
<b>Channel</b>	An open connection between an application on the device (e.g. mobile phone) and an applet on the SE.

## **2. Overview**

This test plan 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.

## 3. 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.

### 3.1 Method: Reader[] getReaders()

#### 3.1.1 Conformance Requirements

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

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

##### Normal execution

CRN1<sup>1</sup>: 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

### 3.2 Method: boolean isConnected ()

#### 3.2.1 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

### 3.3 Method: void shutdown ()

#### 3.3.1 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.

---

<sup>1</sup> CRN stands for Conformance Requirement Number

Parameter errors

None

Context errors

None

### 3.4 Constructor: **SEService(Context context, SEService.CallBack listener)**

#### 3.4.1 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 system.

CRN2: The `isConnected()` method of the specified listener is called if the connection process is finished.

CRN3: The `serviceConnected()` method returns true if the connection process is finished.

Parameter errors

`IllegalArgumentException` - if the parameter context is null.

Context errors

None



## 4. 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.

### 4.1 Method: void serviceConnected(SEService service)

#### 4.1.1 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 framework calls this method when the service is connected. 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

## 5. Class: Reader

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.

### 5.1 Method: String getName()

#### 5.1.1 Conformance Requirements

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

```
String getName()
```

##### Normal execution

CRN1: this method returns the user-friendly name of this reader.

CRN2: If a reader is a SIM reader, then its name must start with the "SIM" prefix.

CRN3: If a reader is a SD or micro SD reader, then its name must start with the "SD" prefix.

CRN4: If a reader is an embedded SE reader, then its name must start with the "eSE" prefix.

##### Parameter errors

None

##### Context errors

None

### 5.2 Method SEService getSEService()

#### 5.2.1 Conformance Requirements

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

```
SEService getSEService ()
```

##### Normal execution

CRN1: Return the service this reader is bound to.

##### Parameter errors

None

##### Context errors

None

### 5.3 Method: boolean isSecureElementPresent()

#### 5.3.1 Conformance Requirements

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

```
boolean isSecureElementPresent()
```

##### Normal execution

CRN1: Checks if a secure element is present in the selected reader. If so, then the method returns true.

CRN2: Checks if a secure element is present in the selected reader. If not, then the method returns false.

Parameter errors

None

Context errors

None

## 5.4 Method: Session openSession()

### 5.4.1 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 a developer to connect to a secure element in the reader

CRN2: the Secure Element needs to be prepared (initialized) for communication (i.e. switched on)

Parameter errors

None

Context errors

IOException - if something went wrong when communicating with the Secure Element or the reader.

## 5.5 Method: void closeSessions()

### 5.5.1 Conformance Requirements

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

```
void closeSession()
```

Normal execution

CRN1: Close 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

## 6. Class: Session

Instances of this class represent a connection session to one of the secure elements available on the device. These objects can be used to get a communication channel with an application in the secure element. This channel can be the basic channel or a logical channel.

### 6.1 Method: Reader getReader()

#### 6.1.1 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

### 6.2 Method: byte[] getATR()

#### 6.2.1 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

### 6.3 Method: void close()

#### 6.3.1 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

## 6.4 Method: boolean isClosed()

### 6.4.1 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

## 6.5 Method: void closeChannels()

### 6.5.1 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

## 6.6 Method: Channel openBasicChannel(byte[] aid)

### 6.6.1 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 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 or Secure Element drivers).

#### Parameter errors

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

**Context errors**

IOException - if something goes wrong with the communication to the reader or the secure element.

IllegalStateException - if the secure element session is used after being closed.

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

**6.7 Method: Channel openLogicalChannel(byte[] aid)****6.7.1 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

**Parameter errors**

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

**Context errors**

IOException - if something goes wrong with the communication to the reader or the secure element. (e.g. if the AID is not available)

IllegalStateException - if the secure element session is used after being closed.

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

## 7. 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. Channels are opened by calling the `Session.openBasicChannel(byte[])` or `Session.openLogicalChannel(byte[])` methods.

### 7.1 Method: void close()

#### 7.1.1 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

### 7.2 Method: boolean isBasicChannel()

#### 7.2.1 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

### 7.3 Method: boolean isClosed()

#### 7.3.1 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

## 7.4 Method: `byte[] getSelectResponse()`

### 7.4.1 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 or the selection response can not be retrieved by the reader implementation.

Parameter errors

None

Context errors

None

## 7.5 Method: `Session getSession()`

### 7.5.1 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

## 7.6 Method: `byte[] transmit(byte[] command)`

### 7.6.1 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 TPDU's 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 TPDU's 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.

### Parameter errors

IllegalParameterError - if the parameter command is null.

IllegalParameterError - if a `MANAGE_CHANNEL` command is sent to the SE

IllegalParameterError - if a `SELECT by DF Name (p1=04)` command is sent to the SE

### Context errors

*IOException - if there is a communication problem to the reader or the Secure Element.*

IllegalStateException - if the channel is closed at the time of invocation of this method

IllegalParameterError - if the command byte array is less than 4 bytes long

IllegalParameterError - if the length of the APDU is not coherent with the length of the command byte array

SecurityError - if the command is filtered by the security policy.

## 8. History

Table 8-1: History

Version	Date	Author	Comment
1.0	3.11.2011	SIMalliance	Initial Release 1.0