Open Mobile API Specification

Second Errata for Version 3.2

Published by simalliance now Trusted Connectivity Alliance

July 2016

Table of Contents

1. Introduction

This document contains all errata notes for Open Mobile API Specification v3.2. All the changes listed in this document will be integrated in the next version of the Open Mobile API specification. This document uses revision marks to show the new applicable content. The document contains only those parts of the subchapters where the errata are made. All the other parts of the Open Mobile API Specification v3.2 remain unchanged and applicable. This document contains all the modifications from the first Errata document for Open Mobile API specification v3.2 released in February 2016.

2. List of modifications

2.1 Chapter 6.1.1 General Rules for Handling of Status Word

Rationale of the errata: In subchapter (b) Using Transport layer T=0 the references to ISO specification are updated using cross referencing instead of using text. The document with reference [10] is

[10] ISO/IEC 7816-3:2006	Identification cards — Integrated circuit cards — Part 3: Cards with contacts — Electrical interface and transmission protocols

Some protocol details are also deleted, as these are specified clearly in the referenced ISO specification.
In subchapter b) under point (i) Specific rules for handling of status word Warning (62xx and 63xx) it is highlighted that the required behaviour when opening a channel is to send GET RESPONSE with Le=00 when receiving 62xx, or 63xx - without data - as the first Response TPDU for the SELECT command. The changes are shown below with tracking mark:

(b) Using Transport layer T=0

Unless otherwise specified, when sending command APDU, SIMalliance's OMAPI Specification requires the following management for status word for all the methods defined in this document:

- For status word '61 XX' the API or underlying implementation shall issue a GET RESPONSE command as specified in [10] ~~by ISO 7816 standard with LE=XX~~. The same behaviour has to be applied for the scenario where the SE returns overall response data of more than 256 bytes to the APDU. If an Error SW is received to GET RESPONSE command, all data received so far shall be discarded and the Error SW shall be returned.
- For the status word '6C XX', the API or underlying implementation shall reissue the input command ~~with LE=XX~~ as specified in [10] ~~by ISO 7816 standard~~. If an Error SW is received to the reissued command, all data received so far shall be discarded and the Error SW shall be returned.

(i) Specific rules for handling of status word Warning (62xx and 63xx)

- In case of openBasicChannel(byte[] aid); openBasicChannel(byte[] aid, byte P2); openLogicalChannel(byte[] aid); openLogicalChannel(byte[] aid, byte P2) and selectNext() the GET RESPONSE with Le=00 shall be sent also in case of SW warning (62xx , 63xx) received as first response for the SELECT command and independently of the setting in *setTransmitBehaviour(boolean expectDataWithWarningSW)* method.

2.2 Chapter 6.1.2 Handling of Event

Rationale of the errata: to clarify the requested behaviour in case of certain Events and correct the name of SEInsertedEvent. The changes are shown below with tracking mark:

(a) IOErrorEvent.

When IOError occurs the API shall implement the following behaviour: Before returning the IOError, the API shall close ALL the opened sessions and channels of ALL applications on the related reader. When all the sessions and channels are closed the API shall asynchronously notify all applications that registered a Reader:EventCallback on the related reader.
If the application wants to continue communicating to the SE after receiving an IOErrorEvent, it should reopen the session and channel.

(b) SEInsert~~ioned~~ed Event and SERemovalEvent.

SEInsert~~ioned~~edEvent and SERemovalEvent occurs upon insertion/removal of SE on a specific reader. With these events, applications can be notified about insertion/removal without polling the SE presence. Before issuing SERemovalEvent the API shall close ALL the opened sessions and channels of ALL applications on the related reader. When all the sessions and channels are closed the API shall asynchronously notify all applications that registered a Reader:EventCallback on the related reader.

2.3 Chapter 6.2.2 e) Method int getVersion()

Rationale of the errata: In order to maintain compatibility with 3.0 and 2.05 versions of OMAPI specification the Method int getVersion() is removed and replaced by String getVersion().The changes are shown below with tracking mark:

(e) Method: ~~int~~ String getVersion()

Returns the version of the Open Mobile API Specification this implementation is based on.

Return value:

~~The version number is returned as an integer value, with the precise type used being one which is natural to the OS platform (with the proviso of a minimum of 32 bits precision).~~
~~The overall version number is calculated based on a major version number and minor version number. The major version number is a value between 2 and 2147483 and the minor number is a value between 0 and 999. The calculation is:~~
~~version_number = (major_version_number * 1000) + minor_version_number (e.g. "3001" for the implementation which is based on the OMAPI specification v3.1).~~
String containing the Open Mobile API version (e.g. "3.2" for Open Mobile API Specification version 3.2).

2.4 Chapter 6.2.5 f) Method: int getEventType

Rationale of the errata: define the exact integer values for the event types and clarify the description of the events. The changes are shown below with tracking mark:

EventType:
0x1001 for Reader:IOErrorEventType – an IOError occurred on the reader
0x2001 for Reader:SEInsertedEventType – the SE was in removed state and has been inserted in the reader
0x2002 for Reader:SERemovalEventType – the SE was in inserted state and has been removed from the reader

2.5 Chapter 6.2.7 f) Method: Channel openBasicChannel(byte[] aid, byte P2)

Rationale of the errata: Fix an editorial error and to highlight that the required behaviour when executing openBasicChannel(byte[] aid, byte P2) *is to send GET RESPONSE with Le=00 when receiving 62xx, or 63xx - without data - as the first Response TPDU for the SELECT command. It applies also to Method: Channel openBasicChannel(byte[] aid). The changes are shown below with tracking mark:*

(f) Method: Channel openBasicChannel(byte[] aid, byte P2)

For the SELECT command, the API shall handle received status word as defined in Chapter 6.1.1, except that for T=0 protocol the GET RESPONSE with Le=00 shall also be sent in case of SW warning received as first response for the SELECT command.

When a ~~logical~~ basic channel is opened the value of the attribute expectDataWithWarningSW shall be 'false'.

2.6 Chapter 6.2.7 h) Method: Channel openLogicalChannel(byte[] aid, byte P2)

Rationale of the errata: To highlight that the required behaviour when executing openBasicChannel(byte[] aid, byte P2) *is to send GET RESPONSE with Le=00 when receiving 62xx, or 63xx - without data - as the first Response TPDU for the SELECT command. It applies also to Method: Channel openBasicChannel(byte[] aid). The changes are shown below with tracking mark:*

(a) Method: Channel openLogicalChannel(byte[] aid, byte P2)

For the SELECT command, the API shall handle received status word as defined in Chapter 6.1.1, except that for T=0 protocol the GET RESPONSE with Le=00 shall also be sent in case of SW warning received as first response of the SELECT command.

2.7 Chapter 6.2.8 f) Method: void setTransmitBehaviour (boolean expectDataWithWarningSW)

Rationale of the errata: To highlight that the behaviour required for transmit() when expectDataWithWarningSW is set to 'true' is to send GET RESPONSE when receiving 62xx, or 63xx - without data - as the first Response TPDU. Also a protocol detail is deleted, as this is an evident behaviour defined by ISO specification and it is not necessary to specify it by Open Mobile API specification. The changes are shown below with tracking mark:

(f) Method: void setTransmitBehaviour (boolean expectDataWithWarningSW)

Sets the expected behaviour of the transmit() method for handling the first received warning SW (62XX,or 63XX). The method applies only for T=0 protocol and only when transmitting case-4 APDU and only for the first received SW warning (62XX, or 63XX). It applies only for this channel object. When opening a new channel object the default value of the expectDataWithWarningSW attribute shall be "false".

Parameter:

expectDataWithWarningSW

- when set to false: the underlying implementation of the transmit() method shall only issue a GET RESPONSE command after receiving a 61XX SW (as defined in Chapter 6.1.1.).
- when set to true: ~~the underlying implementation of the transmit() method shall first cut off the Le field of the command APDU.~~
  - In case the SE returns data and warning SW ('62XX' or '63XX'), the returned data and the warning SW shall be provided to the calling Mobile Application and no GET RESPONSE shall be sent.
  - In case the SE returns a warning SW ('62XX' or '63XX') without data, the API shall issue a GET RESPONSE with Le='00'. After sending the GET RESPONSE with Le='00', general rules specified in Chapter 6.1.1 shall apply for retrieving the data. If all the available data was successfully retrieved - i.e. the last status word is '9000', or a warning status word (62 xx, 63 xx) - the API shall return the data together with the first received warning status word (i.e. the warning status word received for the command APDU sent in the transmit() method). In consequence the last success or warning status word shall be ignored.
    Note: If the SE has no data to provide back after sending the warning SW, the GET RESPONSE will trigger an error on the SE. So, when setting the value to "true", it is strongly recommended to check that the SE applet will have data to provide back to the calling application in case of any warning status word returned by the applet.

2.8 Chapter 6.2.8 g) Method: byte[] transmit(byte[] command)

Rationale of the errata: to align with other industry organisation the requirement to support extended length APDU is mandatory from the 1st of January 2018.
For the API implementation it is still mandatory to be able to handle concatenated APDUs, when the response data is bigger than 256 bytes. The changes are shown below with tracking mark:

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

The transmit method shall support extended length APDU commands from the 1st of January 2018.

2.9 Chapter 6.3.2 a) OMAPI RESULT omapi_get_version

Rationale of the errata: same as for 2.3 in procedural interface. The changes are shown below with tracking mark:

(a) OMAPI RESULT omapi_get_version(String pVersion, Int *pLength~~Int *pVersion~~)

~~pVersion has to be a properly allocated integer.~~
pVersion has to be a properly allocated string or null if the function should return the proper string length in pLength.
Depending on the software platform, pVersion might include a \0 after the version name if the platform representation of strings are zero-terminated.

Parameters
~~[out] Int *pVersion - allocated integer where the version number will be stored.~~
[out] String pVersion - allocated string buffer that will contain the version string or null to determine the proper length in pLength.
[in|out] Int *pLength - size of version string length.

Return
Success - all went ok.
IllegalParameterError - pLength is too short for the version string to be returned.
GeneralError - general error not further specified.

2.10 Chapter 6.3.4 ReaderEvent Type

Rationale of the errata: same as for 2.4 in procedural interface. The changes are shown below with tracking mark:

The event types are:
- 0x1001 for ReaderEvent:IOErrorReaderEventType – an IOError occurred on the reader. This IOError may be triggered by another application.
- 0x2001 for ReaderEvent:SEInsertedReaderEventType, - the SE was ~~not inserted~~in removed state and has been inserted in the reader.

- 0x2002 for ReaderEvent:SERemovedReaderEventType – the SE was in inserted state and has been removed from the reader.

2.11 Chapter 6.3.5 a) OMAPI RESULT omapi_reader_get_name

Rational of the errata: fix an error related to the pReader parameter: pReader refers to the allocated memory and not to the address of the pointer referring to the allocated memory. The changes are shown below with tracking mark:

(a) OMAPI RESULT omapi_reader_get_name(Handle hReader, String ~~*~~pReader, Int *pLength)

pReader must be properly allocated or null where the function returns the proper amount of memory to be allocated in pLength.
Depending on the software platform, pReader ~~pLength~~ might include a \0 after the actual reader name if the platform representation of strings ~~are~~is zero-terminated.

Parameters

[in] Handle hReader - handle to the reader.
[out] String ~~*~~pReader - allocated string buffer that will contain ~~to retrieve~~ the name of the reader or null to determine the string length.
[in|out] Int *pLength - size of allocated/returned string.

2.12 Chapter 6.3.7 f) OMAPI RESULT omapi_channel_set_transmit_behaviour(Handle hChannel, Boolean expectDataWithWarningSW)

Rationale of the errata: Same change as in point 2.7 of this document applied for the procedural interface also. The changes are shown below with tracking mark:

(f) OMAPI RESULT omapi_channel_set_transmit_behaviour(Handle hChannel, Boolean expectDataWithWarningSW)

Parameters

[in] Boolean expectDataWithWarningSW

- when set to false: the underlying implementation of the omapi_channel_transmit() function shall only issue a GET RESPONSE command after receiving a 61XX SW (as defined in Chapter 6.1.1.).
- when set to true: ~~the underlying implementation of the omapi_channel_transmit() function shall first cut off the Le field of the command APDU.~~
  - In case the SE returns data and warning SW ('62XX' or '63XX'), the returned data and the warning SW shall be provided to the calling Mobile Application and no GET RESPONSE shall be sent.

- In case the SE returns a warning SW ('62XX' or '63XX') without data, the API shall issue a GET RESPONSE with Le='00'. After sending the GET RESPONSE with Le='00', general rules specified in Chapter 6.1.1 shall apply for retrieving the data. If all the available data was successfully retrieved - i.e. the last status word is '9000', or a warning status word (62 xx, 63 xx) - the API shall return the data together with the first received warning status word (i.e. the warning status word received for the command APDU sent in the omapi_channel_transmit()). In consequence the last success, or warning status word, shall be ignored.
  Note: If the SE has no data to provide back after sending the warning SW, the GET RESPONSE will trigger an error on the SE. So, when setting the value to "true", it is strongly recommended to check that the SE applet will have data to provide back to the calling application in case of any warning status word returned by the applet.

2.13 Chapter 8 Minimum Set of Functionality

Rationale of the errata: same as for 2.8. The changes are shown below with tracking mark:

The Transport API shall support extended length APDU commands from the 1st of January 2018.

2.14 Annex A: Ansi-C Reference Header for Transport Procedural Interface

Rationale of the errata: update the omapi_get_version and the omapi_reader_get_name functions and one reader event type name. The changes are shown below with tracking mark:

```
/* omapi.h
 * Copyright (c) 2014-6 SIMalliance.org*/
#ifndef __omapi_h__
#define __omapi_h__

#ifdef __cplusplus
extern "C" {
#endif

/* platform specific mapping of SIMalliance data types */
#ifndef OMAPI_API
#define                         OMAPI_API
#endif
typedef int                     OMAPI_RESULT;
```

```
typedef int                     OMAPI_HANDLE;
typedef int                     Int;
typedef char *                  String;
typedef unsigned char           Byte;
typedef enum { false, true }    Boolean;
typedef void (*omapi_reader_eventcallback)(void* callbackcontext,
OMAPI_HANDLE hReader, Int eventtype);

/* SIMalliance return codes */
#define OMAPI_SUCCESS                           ((Int)0x00000000) /* No
error was encountered */
#define OMAPI_GENERAL_ERROR                     ((Int)0x10000000) /* A
general error occurred */
#define OMAPI_IO_ERROR                          ((Int)0x10000001) /*
Communication error */
#define OMAPI_NO_SUCH_ELEMENT_ERROR             ((Int)0x10000002) /* No
such element error */
#define OMAPI_ILLEGAL_STATE_ERROR               ((Int)0x10000003) /*
Illegal state of execution error */
#define OMAPI_ILLEGAL_PARAMETER_ERROR           ((Int)0x10000004) /*
Illegal or invalid parameter */
#define OMAPI_ILLEGAL_REFERENCE_ERROR           ((Int)0x10000005) /*
Illegal reference */
#define OMAPI_OPERATION_NOT_SUPPORTED_ERROR     ((Int)0x10000006) /*
Operation not supported from SE */
#define OMAPI_SECURITY_ERROR                    ((Int)0x10000007) /*
Security Error blocks execution */
#define OMAPI_CHANNEL_NOT_AVAILABLE_ERROR       ((Int)0x10000008) /* No
channel available */
#define OMAPI_NULL POINTER_ERROR                ((Int)0x10000009) /* Null
pointer not allowed */

/* reader event types */
#define OMAPI_READER_IOERROR_EVENTTYPE          ((Int)0x1001)
#define OMAPI_READER_SEINSERTIONED_EVENTTYPE          ((Int)0x2001)
#define OMAPI_READER_SEREMOVAL_EVENTTYPE         ((Int)0x2002)


/* SIMalliance Open Mobile API */
OMAPI_API OMAPI_RESULT omapi_get_readers(OMAPI_HANDLE *phReaders, Int
*pLength);
OMAPI_API OMAPI_RESULT omapi_get_version(Int *pVersion);
OMAPI_API OMAPI_RESULT omapi_get_version(String pVersion, Int *pLength);
OMAPI_API OMAPI_RESULT omapi_reader_get_name(OMAPI_HANDLE hReader, String
*pReader, Int *pLength);
OMAPI_API OMAPI_RESULT omapi_reader_get_name(OMAPI_HANDLE hReader, String
pReader, Int *pLength);
OMAPI_API OMAPI_RESULT omapi_reader_is_secure_element_present(OMAPI_HANDLE
hReader, Boolean *pIsPresent);
OMAPI_API OMAPI_RESULT omapi_reader_open_session(OMAPI_HANDLE hReader,
OMAPI_HANDLE *phSession);
```

```
OMAPI_API OMAPI_RESULT omapi_reader_close_sessions(OMAPI_HANDLE hReader);
OMAPI_API OMAPI_RESULT
omapi_reader_register_reader_eventcallback(OMAPI_HANDLE hReader,
omapi_reader_eventcallback cb, void* callbackcontext);
OMAPI_API OMAPI_RESULT
omapi_reader_unregister_reader_eventcallback(OMAPI_HANDLE hReader,
omapi_reader_eventcallback cb, Boolean* out_unregistered);

OMAPI_API OMAPI_RESULT omapi_session_get_reader(OMAPI_HANDLE hSession,
OMAPI_HANDLE *phReader);
OMAPI_API OMAPI_RESULT omapi_session_get_atr(OMAPI_HANDLE hSession, Byte
*pAtr, Int *pLength);
OMAPI_API OMAPI_RESULT omapi_session_close(OMAPI_HANDLE hSession);
OMAPI_API OMAPI_RESULT omapi_session_is_closed(OMAPI_HANDLE hSession,
Boolean *pIsClosed);
OMAPI_API OMAPI_RESULT omapi_session_close_channels(OMAPI_HANDLE hSession);
OMAPI_API OMAPI_RESULT omapi_session_open_basic_channel(OMAPI_HANDLE
hSession, Byte *AID, Int length, Byte P2, OMAPI_HANDLE *phChannel);
OMAPI_API OMAPI_RESULT omapi_session_open_logical_channel(OMAPI_HANDLE
hSession, Byte *AID, Int length, Byte P2, OMAPI_HANDLE *phChannel);

OMAPI_API OMAPI_RESULT omapi_channel_close(OMAPI_HANDLE hChannel);
OMAPI_API OMAPI_RESULT omapi_channel_is_basic_channel(OMAPI_HANDLE
hChannel, Boolean *pIsBasicChannel);
OMAPI_API OMAPI_RESULT omapi_channel_is_closed(OMAPI_HANDLE hChannel,
Boolean *pIsClosed);
OMAPI_API OMAPI_RESULT omapi_channel_get_select_response(OMAPI_HANDLE
hChannel, Byte *pSelectResponse, Int *pLength);
OMAPI_API OMAPI_RESULT omapi_channel_get_session(OMAPI_HANDLE hChannel,
OMAPI_HANDLE *phSession);
OMAPI_API OMAPI_RESULT omapi_channel_set_transmit_behaviour(OMAPI_HANDLE
hChannel, Boolean expectDataWithWarningSW);
OMAPI_API OMAPI_RESULT omapi_channel_transmit(OMAPI_HANDLE hChannel, Byte
*pCommand, Int cmdLength, Byte *pResponse, Int *pRspLength);
OMAPI_API OMAPI_RESULT
omapi_channel_transmit_retrieve_response(OMAPI_HANDLE hChannel, Byte
*pResponse, Int *pRspLength);
OMAPI_API OMAPI_RESULT omapi_channel_select_next(OMAPI_HANDLE hChannel,
Boolean *pSuccess);



#ifdef __cplusplus
}
#endif

#endif
```