ETSI TS 143 019 V4.1.0 (2001-12)

Technical Specification

Digital cellular telecommunications system (Phase 2+);
Subscriber Identity Module Application Programming Interface (SIM API) for Java Card;
Stage 2
(3GPP TS 43.019 version 4.1.0 Release 4)

Reference
RTS/TSGT-0343019Uv4R1

Keywords
GSM

ETSI

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

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

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

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

Foreword

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

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

The cross reference between GSM, UMTS, 3GPP and ETSI identities can be found under www.etsi.org/key .

Contents

Intellectual Property Rights 2
Foreword 2
Foreword 4
1 Scope 5
2 References 5
3 Definitions and abbreviations 5
3.1 Definitions 5
3.2 Abbreviations 6
4 Description 6
4.1 GSM Java Card Architecture 7
4.2 Java Card Selection Mechanism 8
5 GSM Framework 8
5.1 Overview 8
5.2 GSM file data access 8
5.3 Access control 8
5.4 GSM low Level API 9
6 SIM Toolkit Framework 9
6.1 Overview 9
6.2 Applet Triggering 9
6.3 Registration 13
6.4 Proactive command handling 13
6.5 Envelope response handling 14
6.6 Handler availability 14
6.7 SIM Toolkit Framework behaviour 16
6.8 Usage of ViewHandler and EditHandler 17
7 SIM toolkit applet 17
7.1 Applet Loading 17
7.2 Object Sharing 17
Annex A (normative): Java Card SIM API 18
Annex B (normative): Java Card SIM API identifiers 19
Annex C (normative): SIM API package version management 20
Annex D (informative): Toolkit applet example 21
Annex E (informative): Change history 24
History 25

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP) based on work originally done by the Special Mobile Group (SMG) in ETSI.

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

1 Scope

The present document defines the stage two description of the Subscriber Identity Module Application Programming Interface (SIM API) internal to the SIM.

This stage two describes the functional capabilities and the information flow for the SIM API implemented on the Java Card 2.1 API specification [6].

The present document includes information applicable to network operators, service providers and SIM, server and database manufacturers.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TR 21.905: "Abbreviations and acronyms".

[2] 3GPP TS 51.011: "Specification of the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface".

[3] 3GPP TS 11.14: "Specification of the SIM Application Toolkit for the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface".

[4] 3GPP TS 23.048: "Security Mechanisms for the SIM application toolkit; Stage 2".

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

[6] 3GPP TS 42.019: "Subscriber Identity Module Application Programming Interface (SIM API); Service description; Stage 1".

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

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

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

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

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

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

Applet : An Applet is an application built up using a number of classes which will run under the control of the Java Card virtual machine. Applets designed for smart cards are sometimes referred to as Cardlets.

Bytecode : Machine independent code generated by a Java compiler and executed by the Java interpreter.

Class : The Class is a type that defines the implementation of a particular kind of object. A Class definition defines instance and class variables and methods.

Framework : A framework defines a set of Application Programming Interface (API) classes for developing applications and for providing system services to those applications.

GSM applet : The GSM application conforming to TS 51.011. It might be a Java Card applet or native application.

Java : An object oriented programming language developed by Sun Microsystems designed to be platform independent.

Method : A Method is a piece of executable code that can be invoked, possibly passing it certain values as arguments. Every Method definition belongs to some class.

Object : The principal building block of object oriented programs. Each object is a programming unit consisting of data (variables) and functionality (methods)

Package : A group of classes. Packages are declared when writing a Java Card program

Toolkit applet : Applet loaded onto the SIM card seen by the Mobile as being part of the SIM Toolkit application and containing only the code necessary to run the application. These applets might be downloaded over the radio interface.

Virtual Machine : The part of the Run-time environment responsible for interpreting the bytecode.

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply, in addition to those listed in TR 21.905 [1]:

AC	Application Code
AID	Application Identifier
APDU	Application Protocol Data Unit
API	Application Programming Interface
CAD	Card Acceptance Device
FFS	For Further Study
IFD	Interface Device
JCRE	Java Card™ Run Time Environment
JVM	Java Virtual Machine
ME	Mobile Equipment
MS	Mobile Station
SIM	Subscriber Identity Module
SE	Sending Entity
SMS-CB	Short Message Service – Cell Broadcast
SMS-PP	Short Message Service – Point to Point
USSD	Unstructured Supplementary Services Data
VM	Virtual Machine

4 Description

The present document describes an API for the GSM SIM. This API allows application programmers access to the functions and data described in TS 51.011 [2] and TS 11.14 [3], such that SIM based services can be developed and loaded onto SIMs, quickly and, if necessarily, remotely, after the card has been issued.

This API is an extension to the Java Card 2.1 API [7] based on the Java Card 2.1 Runtime Environment [8].

4.1 GSM Java Card Architecture

The over all architecture of the SIM Toolkit API based on Java Card 2.1 is:

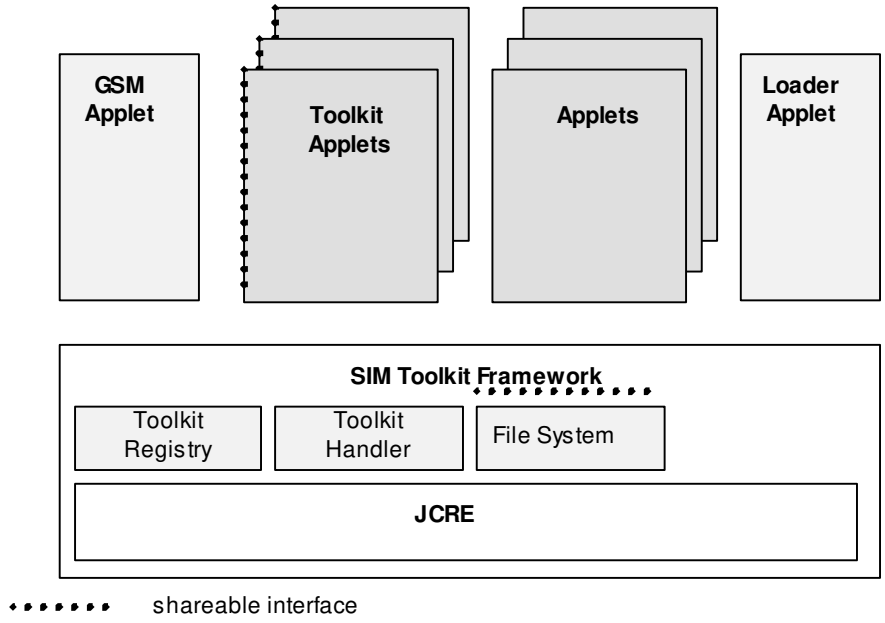

Figure 1: GSM Java Card Architecture

SIM Toolkit Framework: this is the GSM Java Card runtime environment, it is composed of the JCRE, the Toolkit Registry, the Toolkit Handler and the File System.

JCRE: this is specified in Java Card 2.1 Runtime Environment Specification [8] and is able to select any specific applet and transmit to it the process of its APDU.

Toolkit Registry: this is handling all the registration information of the toolkit applets, and their link to the JCRE registry.

Toolkit Handler: this is handling the availability of the system handler and the toolkit protocol (i.e. toolkit applet suspension).

File System: this contains the card issuer file system, and handles the file access control and the applet file context. It is a JCRE owned object implementing the shareable interface *sim.access.SIMView*.

Applets: these derive from javacard.framework.applet and provide the entry points : *process, select, deselect, install* as defined in the Java Card 2.1 Runtime Environment Specification [8].

Toolkit applets: these derive from javacard.framework.applet, so provide the same entry points, and implement the shareable interface *sim.toolkit.ToolkitInterface* so that these applets can be triggered by an invocation of their *processToolkit* method. These applets' AID is defined in TS 101 220 [10].

GSM Applet: this is the default applet as defined in Java Card 2.1 Runtime Environment Specification [8], it behaves as regular applet e.g. when another applet is selected via the SELECT AID APDU its *deselect* method is invoked. It's AID is defined in TS 101 220 [10]. This applet handles the TS 51.011[2] APDUs, CHV1/2, the GSM authentication algorithm and the subscriber file access control according to TS 51.011[2].

Loader applet: this is handling the installation and uninstallation of the applets as specified in the applet loading specification TS 23.048 [4].

Shareable interface: this is defined in the Java Card 2.1 specifications.

4.2 Java Card Selection Mechanism

The Java Card selection mechanism is defined in the Java Card Runtime Environment Specification [8].

5 GSM Framework

5.1 Overview

The GSM Framework consists of the GSM applet and the JCRE File System Object.

The GSM Framework is based on two packages:

- The GSM low level package [FFS];
- The *sim.access* package, which allows applets to access the GSM files.

5.2 GSM file data access

The following methods shall be offered by the API to card applets, to allow access to the GSM data:

select	Select a file without changing the current file of any other applet or of the subscriber session. At the invocation of the *processToolkit* method of a toolkit applet, the current file is the MF. The toolkit applet file context remains unchanged during the whole execution of the *processToolkit* method, the current record may be altered if the current file is a cyclic file and the content of the current file may be altered. This method returns the selected file information;
status	Read the file status information of the current DF;
readBinary	Read data bytes of the transparent EF currently selected by the applet;
readRecord	Read data bytes of the linear fixed or cyclic EF currently selected by the applet without changing the current record pointer of any other applet / subscriber. This method allows reading part of a record;
updateBinary	Modify data bytes of the transparent EF currently selected by the applet. The toolkit applet shall send the corresponding refresh ;
updateRecord	Modify data bytes of the linear fixed or cyclic EF currently selected by the applet. The current record pointer of other applets / subscriber shall not be changed in case of linear fixed EF but the record pointer of a cyclic EF shall be changed for all other applets / subscriber to the record number 1. This method allows updating part of a record. The toolkit applet shall send the corresponding refresh ;
seek	Search a record of the linear fixed file currently selected by the applet starting with a given pattern. The current record pointer of any other applet or of the subscriber session shall not be changed;
increase	Increase the value of the last updated record of the cyclic EF currently selected. It becomes than record number 1 for every other applet and subscriber session. This method returns the increased value. The toolkit applet shall send the corresponding refresh;
rehabilitate	Rehabilitate the EF currently selected by the applet with effect for all other applets / subscriber. The toolkit applet shall send the corresponding refresh;
invalidate	Invalidate the EF currently selected by the applet with effect for all other applets / subscriber. The toolkit applet shall send the corresponding refresh.

These methods are described in the *sim.access.SIMView* interface in Annex A.

5.3 Access control

The Access Control privileges of the applet are granted during installation according to the level of trust. When an applet requests access to GSM or operator specific files, the SIM Toolkit Framework checks if this access is allowed by examination of the file control information stored on the card. If access is granted the SIM Toolkit Framework will process the access request, if access is not granted, an exception will be thrown.

[Contents and coding of the file(s) containing access control information will be defined in TS 51.011]

5.4 GSM low Level API

[FFS. This API allows the implementation of the GSM applet]

6 SIM Toolkit Framework

6.1 Overview

The SIM API shall consist of APIs for TS 11.14 [3] (pro-active functions) and TS 51.011 [2] (transport functions).

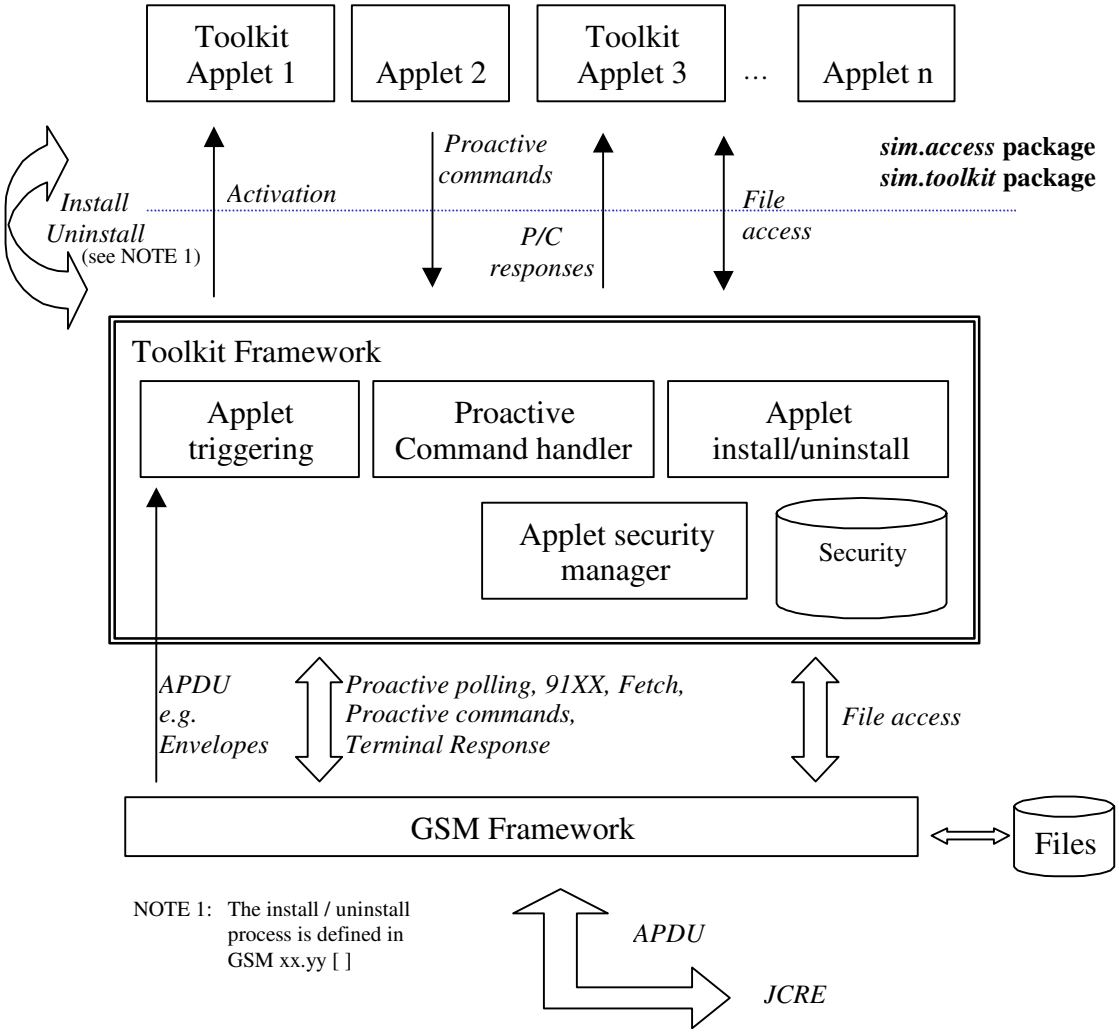

Figure 2: SIM Toolkit Framework functional description

In this model, the GSM data field structure is viewed as a series of data objects to the API. In the physical model of course, they may still be stored in elementary fields, but classes will access these data as part of the objects within those classes.

6.2 Applet Triggering

The application triggering portion of the SIM Toolkit Framework is responsible for the activation of toolkit applets, based on the APDU received by the GSM application.

Figure 3: toolkit applet triggering diagram

The ME shall not be adversely affected by the presence of applets on the SIM card. For instance a syntactically correct Envelope shall not result in an error status word in case of a failure of an applet. The only application as seen by the ME is the SIM application. As a result, a toolkit applet may throw an exception, but this error will not be sent to the ME.

The difference between a Java Card applet and a Toolkit applet is that the latter does not handle APDUs directly. It will handle higher level messages. Furthermore the execution of a method could span over multiple APDUs, in particular, the proactive protocol commands (Fetch, Terminal Response).

As seen above, when the GSM applet is the selected application and when a toolkit applet is triggered the *select*() method of the toolkit applet shall not be launched since the toolkit applet itself is not really selected.

Here after are the events that can trigger a toolkit applet :

EVENT_PROFILE_DOWNLOAD

Upon reception of the Terminal Profile command by the SIM, the SIM Toolkit Framework stores the ME profile and then triggers the registered toolkit applet which may want to change their registry. A toolkit applet may not be able to issue a proactive command.

EVENT_MENU_SELECTION, EVENT_MENU_SELECTION_HELP_REQUEST

A toolkit applet might be activated upon selection in the ME's menu by the user, or request help on this specific menu.

In order to allow the user to choose in a menu, the SIM Toolkit Framework shall have previously issued a SET UP MENU proactive command. When a toolkit applet changes a menu entry of its registry object, the SIM Toolkit Framework shall dynamically update the menu stored in the ME during the current card session. The SIM Toolkit Framework shall use the data of the EFsume file when issuing the SET UP MENU proactive command.

The positions of the toolkit applet menu entries in the item list, the requested item identifiers and the associated limits (e.g. maximum length of item text string) are defined at the loading of the toolkit applet.

If at least one toolkit applet registers to EVENT_MENU_SELECTION_HELP_REQUEST, the SET UP MENU proactive command sent by the SIM Toolkit Framework shall indicate to the ME that help information is available.A toolkit applet registered for one or more menu entries, may be triggered by the event EVENT_MENU_SELECTION_HELP_REQUEST, even if it is not registered to this event. A toolkit applet registered for one or more menu entries should provide help information.

EVENT_FORMATTED_SMS_PP_ENV, EVENT_UNFORMATTED_SMS_PP_ENV, EVENT_FORMATTED_SMS_PP_UPD, EVENT_UNFORMATTED_SMS_PP_UPD

A toolkit applet can be activated upon the reception of a short message.

There are two ways for a card to receive an SMS : via the Envelope SMS-PP Data Download or the Update Record EFsms instruction.

The reception of the SMS by the toolkit applet cannot be guaranteed for the Update Record EFsms instruction.

The received SMS may be :
- formatted according to TS 23.048[4] or an other protocol to identify explicitly the toolkit applet for which the message is sent ;
- unformatted or using a toolkit applet specific protocol the SIM Toolkit Framework will pass this data to all registered toolkit applets.

EVENT_FORMATTED_SMS_PP_ENV

This event is triggered by an envelope APDU containing an SMS_DATADOWNLOAD BER TLV with an SMS_TPDU simple TLV according to TS 23.048[4].

The SIM Toolkit Framework shall:
- verify the TS 23.048[4] security of the SMS TPDU ;
- trigger the toolkit applet registered with the corresponding TAR defined at applet loading;
- take the optional Application Data posted by the triggered toolkit applet if present;
- secure and send the response packet.

The toolkit applet will only be triggered if the TAR is known and the security verified, application data will also be deciphered.

EVENT_UNFORMATTED_SMS_PP_ENV

The registered toolkit applets will be triggered by this event and get the data transmitted in the APDU envelope SMS_DATADOWNLOAD.

But only the first toolkit applet triggered will be able to send back a response as defined by the rules in chapter 6.6.

EVENT_FORMATTED_SMS_PP_UPD

This event is triggered by Update Record EFsms with an SMS TP-UD field formatted according to TS 23.048[4].

The SIM Toolkit Framework shall :
- update the EFsms file with the data received, it is then up to the receiving toolkit applet to change the SMS stored in the file (i.e. the toolkit applet need to have access to the EFsms file)
- verify the TS 23.048[4] security of the SMS TPDU ;
- convert the Update Record EFsms in a TLV List, an EnvelopeHandler ;
- trigger the toolkit applet registered with the corresponding TAR defined at applet loading;

The Update Record EFsms APDU shall be converted in a TLV list as defined below :

UPDATE RECORD APDU	nb bytes	Handler TLV LIST	size
CLA, INS	2	specific event	1
P1,P2	2	device Identity rec-number	1
P3 = 176	1		1
status	1	device Identity rec-status	1
TS-SCA (RP-OA)	<= 12	Address	Y
SMS TPDU	var	SMS TPDU	Y
padding bytes	var		Y

The EnvelopeHandler provided to the applet shall:
- return *BTAG_SMS_PP_DOWNLOAD* to the *getEnvelopeTag()* method call**;**
- return the Simple TLV list length to the *getLength()* method call;
- contain the Simple TLV list :

EnvelopeHandler TLV List
Device identities
Address
SMS TPDU

The applet should use the *findTLV()* methods to get each Simple TLV.

The Device Identity Simple TLV is used to store the information about the absolute record number in the EFsms file and the value of the EFsms record status byte, and formatted as defined below:

Device identities Simple TLV
Device identities tag
length = 02
Absolute Record Number
Record Status

With the absolute record number the toolkit applet can update EFsms in absolute mode to change the received SMS in a readable text.

EVENT_UNFORMATTED_SMS_PP_UPD

The SIM Toolkit Framework will first update the EFsms file, convert the received APDU as described above, and then trigger all the registered toolkit applets. All of them may modify the content of EFsms (i.e. the toolkit applets need to have access to the EFsms file).

EVENT_FORMATTED_SMS_CB, EVENT_UNFORMATTED_SMS_CB

When the ME receives a new cell broadcast message, the cell broadcast page may be passed to the SIM using the envelope command according to the content of the EF_{CBMID} file. E.g. the application may then read the message and extract a meaningful piece of information which could be displayed to the user, for instance.

The received cell broadcast page can be either:
- formatted according to TS 23.048 [4] or an other protocol to identify explicitly the toolkit applet for which the message is sent ;
- unformatted or using a toolkit applet specific protocol the SIM Toolkit Framework will pass this data to all registered toolkit applets.

EVENT_FORMATTED_SMS_CB

This event is triggered by an envelope APDU containing an CELL_BROADCAST_DATADOWNLOAD BER TLV with a Cell Broadcast Page simple TLV according to TS 23.048 [4].

The SIM Toolkit Framework shall:
- verify the TS 23.048[4] security of the Cell Broadcast Page;
- trigger the toolkit applet registered with the corresponding TAR defined at applet loading.

The toolkit applet will only be triggered if the TAR is known and the security verified, application data will also be deciphered.

The TAR value is the same as the one used in the events *EVENT_FORMATTED_SMS_PP_ENV* and *EVENT_FORMATTED_SMS_PP_UPD.*

EVENT_UNFORMATTED_SMS_CB

The registered toolkit applets will be triggered by this event and get the data transmitted in the APDU envelope CELL_BROADCAST_DATADOWNLOAD.

EVENT_CALL_CONTROL_BY_SIM

When the SIM is in call control mode and when the user dials a number, this number is passed to the SIM. Only one toolkit applet can handle the answer to this command: call barred, modified or accepted.

EVENT_EVENT_DOWNLOAD_MT_CALL, EVENT_EVENT_DOWNLOAD_CALL_CONNECTED, EVENT_EVENT_DOWNLOAD_CALL_DISCONNECTED, EVENT_EVENT_DOWNLOAD_LOCATION_STATUS, EVENT_EVENT_DOWNLOAD_USER_ACTIVITY, EVENT_EVENT_DOWNLOAD_IDLE_SCREEN_AVAILABLE, EVENT_EVENT_DOWNLOAD_CARD_READER_STATUS, EVENT_EVENT_DOWNLOAD_LANGUAGE_SELECTION, EVENT_EVENT_DOWNLOAD_BROWSER_TERMINATION

The toolkit applet will be triggered by the registered event download trigger, upon reception of the corresponding Envelope command.

In order to allow the toolkit applet to be triggered by these events, the SIM Toolkit Framework shall have previously issued a SET UP EVENT LIST proactive command. When a toolkit applet changes one or more of

these requested events of its registry object, the SIM Toolkit Framework shall dynamically update the event list stored in the ME during the current card session.

EVENT_MO_SHORT_MESSAGE_CONTROL_BY_SIM

Before sending an SMS MO entered by the user, the SMS is submitted to the SIM. Only one toolkit applet can register to this event

EVENT_TIMER_EXPIRATION

At the registration to this event the toolkit applet gets the reference to its timer. The toolkit applet can then manage the timer, it will be triggered at the reception of the APDU Envelope TIMER EXPIRATION.

The SIM Toolkit Framework shall reply busy to this Envelope APDU if it cannot guaranty to trigger the corresponding toolkit applet.

EVENT_UNRECOGNIZED_ENVELOPE

The applet registered to this event shall be triggered by the framework if the BER-TLV tag contained in the ENVELOPE APDU is not defined in the associated release of TS 11.14 [3] and if no corresponding constant is defined in the list of the ToolkitConstants interface. The unrecognized Envelope event will allow a toolkit applet to handle the evolution of the TS 11.14 specification.

EVENT_STATUS_COMMAND

At reception of a STATUS APDU command, the SIM Toolkit Framework shall trigger the registered toolkit applet.

A range of events is reserved for proprietary usage (from –128 to –1). The use of these events will make the toolkit applet incompatible.

The toolkit applet shall be triggered for the registered events upon reception, and shall be able to access to the data associated to the event using the methods provided by the *sim.toolkit.ViewHandler.EnvelopeHandler* class.

The order of triggering the toolkit applet shall follow the priority level of each toolkit applet defined at its loading. If several toolkit applets have the same priority level, the last loaded toolkit applet takes precedence.

6.3 Registration

During it's installation the toolkit applet shall register to the JCRE and the SIM Toolkit Framework so that it can be triggered by both selection mechanisms.

The toolkit applet will have to call the *getEntry()* method to get a reference to it's registry and then to explicitly register to each event it requires.

The toolkit applet can change the events to which it is registered during its life cycle.

The toolkit applet will dynamically register itself to some event e.g. *EVENT_MENU_SELECTION* by calling the corresponding method e.g. *initMenuEntry()*.

The API is described in the *sim.toolkit.ToolkitRegistry* class in Annex A.

6.4 Proactive command handling

The SIM application toolkit protocol (i.e. 91xx, Fetch, Terminal Response) is handled by the GSM applet and the Toolkit Handler, the toolkit applet shall not handle those events.

The SIM Toolkit Framework shall provide a reference of the *sim.toolkit.ViewHandler.EditHandler.ProactiveHandler* to the toolkit applet so that when the toolkit applet is triggered it can :

- initialise the current proactive command with the *init()* method ;
- append several Simple TLV as defined in TS 11.14 [3] to the current proactive command with the *appendTLV()* methods ;

- ask the SIM Toolkit Framework to send this proactive command to the ME and wait for the reply, with the *send()* method.

The GSM applet and the SIM Toolkit Framework shall handle the transmission of the proactive command to the ME, and the reception of the response. The SIM Toolkit Framework will then return in the toolkit applet just after the *send()* method. It shall then provide to the toolkit applet the *sim.toolkit.ViewHandler.ProactiveResponseHandler*, so that the toolkit applet can analyse the response.

The proactive command is sent to the ME as defined and constructed by the toolkit applet without any check of the SIM Toolkit Framework.

The toolkit applet shall not issue the following proactive commands : SET UP MENU, SET UP EVENT LIST, POLL INTERVAL, POLLING OFF ; as those are system proactive commands that will affect the services of the SIM Toolkit Framework.

The SIM Toolkit Framework cannot guarantee that if the SET UP IDLE MODE TEXT proactive command is used by a toolkit applet, another toolkit applet will not overwrite this text at a later stage.

6.5 Envelope response handling

To allow a toolkit applet to answer to some specific events (e.g. EVENT_CALL_CONTROL_BY_SIM) the SIM Toolkit Framework shall provide the *sim.toolkit.ViewHandler.EditHandler.EnvelopeResponseHandler*.

The toolkit applet can then post a response to some events with the *post()* or the *postAsBERTLV()* methods, the toolkit applet can continue it's processing (e.g. prepare a proactive command) the SIM Toolkit Framework will return the response APDU defined by the toolkit applet (i.e. 9F xx or 9E xx).

6.6 Handler availability

The system handlers : ProactiveHandler, ProactiveResponseHandler, EnvelopeHandler and EnvelopeResponseHandler are Temporary JCRE Entry Point Object as defined in the Java Card Runtime Environment Specification [8].

The following table describes the minimum availability of the handlers for all the events at the invocation of the processToolkit method of the toolkit applet.

Table 1: Handler availability for each event

EVENT_	Reply busy	ProactiveHandler ProactiveResponseHandler	EnvelopeHandler	EnvelopeResponseHandler	Nb of triggered / registrered Applet
_FORMATTED_SMS_PP_ENV	Y	Y	Y	Y	1 / n (per TAR)
_FORMATTED_SMS_PP_UPD	N	Y	Y	N	1 / n (per TAR)
_UNFORMATTED_SMS_PP_ENV	Y	Y	Y	Y	n / n
_UNFORMATTED_SMS_PP_UPD	N	Y	Y	N	n / n
_FORMATTED_SMS_CB	Y	Y	Y	N	1 / n (per TAR)
_UNFORMATTED_SMS_CB	Y	Y	Y	N	n / n
_MENU_SELECTION	Y	Y	Y	N	1 / n (per Item Id)
_MENU_SELECTION_HELP_REQUEST	Y	Y	Y	N	1 / n (per Item Id)
_CALL_CONTROL	N	Y/N (see Note 2)	Y	Y	1 / 1
_SMS_MO_CONTROL	N	Y/N (see Note 2)	Y	Y	1 / 1
_TIMER_EXPIRATION	Y	Y	Y	N	1/ 8 (per timer) (see Note 1)
_EVENT_DOWNLOAD					
_MT_CALL	Y	Y	Y	N	n / n
_CALL_CONNECTED	Y	Y	Y	N	n / n
_CALL_DISCONNECTED	Y	Y	Y	N	n / n
_LOCATION_STATUS	Y	Y	Y	N	n / n
_USER_ACTIVITY	Y	Y	Y	N	n / n
_IDLE_SCREEN_AVAILABLE	Y	Y	Y	N	n / n
_LANGUAGE_SELECTION	Y	Y	Y	N	n / n
_BROWSER_TERMINATION	Y	Y	Y	N	n / n
_CARD_READER_STATUS	Y	Y	Y	N	n / n
_UNRECOGNISED_ENVELOPE	Y	Y	Y	Y	n / n
_STATUS_COMMAND	N	Y/N (see Note 2)	N	N	n / n
_PROFILE_DOWNLOAD	N	Y/N (see Note 2)	N	N	n / n

NOTE 1: One toolkit applet can register to several timers, but a timer can only be allocated to one toolkit applet.
NOTE 2: Y/N means that handlers may / may not be available depending whether a proactive session is ongoing.

The following rules define the minimum requirement for the availability of the system handlers and the lifetime of their content.

ProactiveHandler:

- The ProactiveHandler is valid from the invocation to the termination of the processToolkit method.
- If a proactive command is pending the ProactiveHandler may not be available.
- At the processToolkit method invocation the TLV-List is cleared.
- At the call of it's init method the content is cleared and then initialised.
- After a call to ProactiveHandler.send method the handler will remain unchanged (i.e. previously send proactive command) until the ProactiveHandler.init or appendTLV methods are called.

ProactiveResponseHandler:

- The ProactiveResponseHandler may not be available before the first call to ProactiveHandler.send method, if available the content is cleared.
- The ProactiveResponseHandler is available after the first call to the ProactiveHandler.send method to the termination of the processToolkit method.
- If a proactive command is pending the ProactiveResponseHandler may not be available.
- The ProactiveResponseHandler content is changed after the call to ProactiveHandler.send method and remains unchanged until next call to the ProactiveHandler.send method.

EnvelopeHandler:

- The EnvelopeHandler and its content are available for all triggered toolkit applets (see Table1), from the invocation to the termination of their processToolkit method.

- The SIM Toolkit Framework guarantees that all registered toolkit applet are triggered and receive the data.

EnvelopeResponseHandler:

- The EnvelopeResponseHandler is available for all triggered toolkit applets, until a toolkit applet has posted an envelope response or sent a proactive command. After a call to the post method the handler is no longer available.

- The EnvelopeResponseHandler content must be posted before the first invocation of a ProactiveHandler.send method or before the termination of the processToolkit, so that the GSM applet can offer these data to the ME (eg 9Fxx/9Exx). After the first invocation of the ProactiveHandler.send method the EnvelopeResponseHandler is no more available.

The following diagram illustrates these rules.

Applet		Applet 1							Applet 2			
method invocation	*processToolkit*	*post*		*init*		*termination*		*init*		*init*		
		init		*send*		*send*		*processToolkit*		*send*		
Envelope Handler												
EnvelopeResponseHandler												
ProactiveHandler												
Proactive ResponseHandler												

Figure 5: Typical handler availability for toolkit applets (see Table 1 for detail)

6.7 SIM Toolkit Framework behaviour

The following rules define the SIM Toolkit Framework behaviour for :

- Triggering of a toolkit applet (invocation of the *processToolkit()* method from the *ToolkitInterface* shareable interface) :
  - The current context is switched to the toolkit applet .
  - A pending transaction is aborted.
  - There is no invocation of the *select()* or the *deselect()* methods.
  - The CLEAR_ON_DESELECT transient object can not be accessed and not created as defined in Java Card 2.1 Runtime Environment Specification [8], as the current selected application is unchanged (eg GSM applet) and does not correspond to the current context which is the toolkit applet.
  - The current file context of the toolkit applet is the MF.
  - The current file context of the current selected applet is unchanged.
  - The toolkit applet cannot access the APDU object.
- Termination of a toolkit applet (return from the *processToolkit()* method):
  - The JCRE switches back to the context of the current selected applet, the GSM applet.
  - There is no invocation of the *select()* or the *deselect()* methods.
  - A pending toolkit applet transaction is aborted.

- The transient data are unchanged.
- The current file context of the toolkit applet is lost.
- The current file context of the current selected applet is unchanged.
- The GSM applet shall not rely on the APDU object content. The APDU content may be changed by the system [For Further Study as the interface between the toolkit system and the GSM applet is not defined yet]

- Invocation of *ProactiveHandler.send()* method :
  - During the execution there might be other context switches, but at the return of the *send()* method the toolkit applet context is restored.
  - There is no invocation of the *select()* or the *deselect()* methods.
  - A pending toolkit applet transaction at the method invocation is aborted.
  - The current file context of the toolkit applet is unchanged (see chapter 5.2). The *send()* method will never return if the GSM applet is deselected and another applet is explicitly selected.
- Emission of system proactive commands (SIM Toolkit framework dynamic behaviour)
  - The SIM Toolkit Framework shall send its system proactive command as soon as no proactive session is pending and all the applets registered to the current events have been triggered and have returned from the processToolkit method invocation.

6.8 Usage of ViewHandler and EditHandler

The ViewHandler and EditHandler classes have been defined to group the properties of the system handler, and may be used in the future to provide a simple mechanism to the toolkit applet to handle TLV lists.

7 SIM toolkit applet

7.1 Applet Loading

The SIM API card shall be compliant to the Java Card 2.1 VM Architecture Specification [9] and to the Annex B to guarantee interoperability at byte code Level.

The applet loading mechanism, protocol and applet life cycle are defined in TS 23.048 [4]

7.2 Object Sharing

The sharing mechanism defined in Java Card 2.1 API Specification [7] and Java Card 2.1 Runtime Environment Specification [8] shall be used by the applet to share data.

The byte parameter of the *getShareableInterfaceObject()* method shall be set to zero (i.e. '00') when the *ToolkitInterface* reference is required.

Annex A (normative): Java Card SIM API

The attached files "Annex_A_java.zip" and "Annex_A_HTML.zip" contains source files for the Java Card SIM API.

Annex B (normative): Java Card SIM API identifiers

The attached file "Annex_B_Export_files.zip" contains source files for the Java Card SIM API identifiers.

NOTE: The export files in this annex have been generated with the following steps and tools :

- Compilation from the API java source file (.java) to the API class files (.class) with the Java compiler from the Java Development Kit version 1.2.2.
- Convertion from the API class files (.class) to the API export files (.exp) with the Java Card 2.1.2 Class File Converter (version 1.2) and the Java Development Kit 1.2.2.

Annex C (normative): SIM API package version management

The following table describes the relationship between each TS 03.19 / TS 43.019 specification version and its SIM API packages AID and Major, Minor versions defined in the export files.

TS 03.19 / 43.019 version	sim.access package		sim.toolkit package	
	AID	Major, Minor	AID	Major, Minor
7.0.0	A000000009 0003FFFFFFFF8910700001	1.0	A000000009 0003FFFFFFFF8910700002	1.0
7.1.0	A000000009 0003FFFFFFFF8910710001	2.0	A000000009 0003FFFFFFFF8910710002	2.0
7.2.0	A000000009 0003FFFFFFFF8910710001	2.0	A000000009 0003FFFFFFFF8910710002	2.0
7.3.0	A000000009 0003FFFFFFFF8910710001	2.0	A000000009 0003FFFFFFFF8910710002	2.0
7.4.0	A000000009 0003FFFFFFFF8910710001	2.1	A000000009 0003FFFFFFFF8910710002	2.1
7.5.0	A000000009 0003FFFFFFFF8910710001	2.1	A000000009 0003FFFFFFFF8910710002	2.1
8.0.0	A000000009 0003FFFFFFFF8910710001	2.2	A000000009 0003FFFFFFFF8910710002	2.2
8.1.0	A000000009 0003FFFFFFFF8910710001	2.2	A000000009 0003FFFFFFFF8910710002	2.2
8.2.0	A000000009 0003FFFFFFFF8910710001	2.2	A000000009 0003FFFFFFFF8910710002	2.2
4.0.0	A000000009 0003FFFFFFFF8910710001	2.2	A000000009 0003FFFFFFFF8910710002	2.2
4.1.0	A000000009 0003FFFFFFFF8910710001	2.2	A000000009 0003FFFFFFFF8910710002	2.2

The package AID coding is defined in TS 101 220 [10]. The SIM API packages' AID are not modified by changes to Major or Minor Version.

The Major Version shall be incremented if a change to the specification introduces byte code incompatibility with the previous version.

The Minor Version shall be incremented if a change to the specification does not introduce byte code incompatibility with the previous version.

Annex D (informative): Toolkit applet example

```
/**
 * Example of Toolkit Applet
 */

package ToolkitAppletExample;

import sim.toolkit.*;
import sim.access.*;
import javacard.framework.*;


public class MyToolkitApplet extends javacard.framework.Applet implements ToolkitInterface,
ToolkitConstants{

    public static final byte MY_INSTRUCTION          = (byte)0x46;
    public static final byte SERVER_OPERATION        = (byte)0x0F;
    public static final byte CMD_QUALIFIER           = (byte)0x80;
    public static final byte EXIT_REQUESTED_BY_USER  = (byte)0x10;
    private byte[] menuEntry =    {(byte)'S',(byte)'e',(byte)'r',(byte)'v',(byte)'i',(byte)'c',
                                   (byte)'e', (byte)'1'};
     private byte[] menuTitle=    {(byte)'M',(byte)'y',(byte)'M',(byte)'e',(byte)'n' ,(byte)'u'};
    private byte[] item1 =        {(byte)'I',(byte)'T',(byte)'E',(byte)'M',(byte)'1' };
    private byte[] item2 =        {(byte)'I',(byte)'T',(byte)'E',(byte)'M',(byte)'2' };
    private byte[] item3 =        {(byte)'I',(byte)'T',(byte)'E',(byte)'M',(byte)'3' };
    private byte[] item4 =        {(byte)'I',(byte)'T',(byte)'E',(byte)'M',(byte)'4' };
    private Object[] ItemList =   { item1, item2, item3, item4 };
    private byte[] textDText =    {(byte)'H',(byte)'e',(byte)'l',(byte)'l',(byte)'o',(byte)' ',
                                   (byte)'w',(byte)'o',(byte)'r',(byte)'l',(byte)'d',(byte)'2'};
    private byte[] textGInput =    {(byte)'Y',(byte)'o',(byte)'u',(byte)'r',(byte)' ',(byte)'n',
                                   (byte)'a',(byte)'m',(byte)'e',(byte)'?'};

    private byte[]  baGSMAID =
{(byte)0xA0,(byte)0x00,(byte)0x00,(byte)0x09,(byte)0x00,(byte)0x01};
    private ToolkitRegistry reg;
    private SIMView gsmFile;
    private byte buffer[] = new byte[10];
    private byte itemId;
    private byte result;
    private boolean repeat;

    /**
     * Constructor of the applet
     */
    public MyToolkitApplet() {

        // get the GSM application reference
        gsmFile = SIMSystem.getTheSIMView();

        // register to the SIM Toolkit Framework
        reg = ToolkitRegistry.getEntry();

        // Define the applet Menu Entry and register to the EVENT_MENU_SELECTION
        itemId = reg.initMenuEntry(menuEntry, (short)0x0000, (short)menuEntry.length,
                                   PRO_CMD_DISPLAY_TEXT, false, (byte) 0x00, (short) 0x0000);
        // register to the EVENT_UNFORMATTED_SMS_PP_ENV
        reg.setEvent(EVENT_UNFORMATTED_SMS_PP_ENV);
    }

    /**
     * Method called by the JCRE at the installation of the applet
     */
    public static void install(byte bArray[], short bOffset, byte bLength) {
        MyToolkitApplet MyApplet = new MyToolkitApplet ();
        MyApplet.register();
    }

    /**
     * Method called by the GSM Framework
     */
    public Shareable getShareableInterfaceObject ( AID clientAID, byte parameter)
    {
```

```
        if (parameter == (byte) 0x00)
        {
            if ( clientAID.partialEquals(baGSMAID, (byte) 0x00, (byte) baGSMAID.length) == true )
                    return ((Shareable) this);
        }
        return(null);
    }

     /**
      * Method called by the SIM Toolkit Framework
      */
     public void processToolkit(byte event) {

        // get the handler references
        EnvelopeHandler          envHdlr = EnvelopeHandler.getTheHandler();
        ProactiveHandler         proHdlr = ProactiveHandler.getTheHandler();
        ProactiveResponseHandler rspHdlr;

         switch(event) {
            case EVENT_MENU_SELECTION:
                // Prepare the Select Item proactive command
                proHdlr.init(PRO_CMD_SELECT_ITEM,(byte)0x00,DEV_ID_ME);
                // Append the Menu Title
                proHdlr.appendTLV((byte) (TAG_ALPHA_IDENTIFIER | TAG_SET_CR),
                                  menuTitle,(short)0x0000,(short)menuTitle.length);
                // add all the Item
                for (short i=(short) 0x0000; i<(short) 0x0004; i++) {
                    proHdlr.appendTLV((byte) (TAG_ITEM | TAG_SET_CR),(byte) (i+1),
                                      (byte[])ItemList[i],(short) 0x0000,
                                      (short)((byte[])ItemList[i]).length);
                }
                // ask the SIM Toolkit Framework to send the proactive command and check the result
                if ((result = proHdlr.send()) == RES_CMD_PERF){
                    rspHdlr = ProactiveResponseHandler.getTheHandler();
                    // SelectItem response handling
                    switch (rspHdlr.getItemIdentifier()) {
                        case 1:
                        case 2:
                        case 3: // DisplayText
                            proHdlr.init(PRO_CMD_DISPLAY_TEXT, CMD_QUALIFIER,
                                         DEV_ID_DISPLAY);
                            proHdlr.appendTLV((byte)(TAG_TEXT_STRING| TAG_SET_CR), DCS_8_BIT_DATA,
                                         textDText,(short)0x0000, (short)textDText.length);
                            proHdlr.send();
                            break;
                        case 4: // Ask the user to enter data and display it
                            do {
                                repeat = false;
                                try {

                                    // GetInput asking the users name
                                    proHdlr.initGetInput((byte)0x01, DCS_8_BIT_DATA,
textGInput,(byte)0x00,
                                        (short)textGInput.length,(short)0x0001,(short)0x0002);
                                    proHdlr.send();

                                    // display the entered text
                                    rspHdlr.copyTextString(textDText,(short)0x0000);
                                    proHdlr.initDisplayText((byte)0x00,DCS_8_BIT_DATA, textDText,
                                        (short)0x0000,(short) textDText.length);
                                    proHdlr.send();
                                }
                                catch (ToolkitException MyException) {
                                    if (MyException.getReason() ==
ToolkitException.UNAVAILABLE_ELEMENT) {
                                        if (rspHdlr.getGeneralResult() != EXIT_REQUESTED_BY_USER)
                                            repeat = true;
                                        break;
                                    }
                                }
                            }
                            while (repeat);
                            break;
                    }
                }
                break;

            case EVENT_UNFORMATTED_SMS_PP_ENV:
```

```
                // get the offset of the instruction in the TP-UD field
                short TPUDOffset = (short) (envHdlr.getTPUDLOffset() + SERVER_OPERATION);

                // start the action requested by the server
                switch (envHdlr.getValueByte((short)TPUDOffset) ) {
                case 0x41 : // Update of a gsm file
                    // get the data from the received SMS
                    envHdlr.copyValue((short)TPUDOffset+1,buffer, (short)0x0000,(short)0x0003);
                    // write these data in the EFpuct
                    gsmFile.select(SIMView.FID_DF_GSM);
                    gsmFile.select(SIMView.FID_EF_PUCT);
                    gsmFile.updateBinary((short)0x0000,buffer,(short)0x0000,(short)0x0003);

                    break;

                case 0x36 : // change the MenuTitle for the SelectItem
                    envHdlr.copyValue((short)TPUDOffset+1, menuTitle,(short)0x0000,(short)0x0006);
                    break;
                }
                break;
            }

    }

     /**
      * Method called by the JCRE, once selected
      */
     public void process(APDU apdu) {
         // Handle the Select AID apdu
         if (selectingApplet()) return;

         switch(apdu.getBuffer()[1]) {
            // specific APDU for this applet to configure the MenuTitle from SelectItem
            case (byte)MY_INSTRUCTION:
                if (apdu.setIncomingAndReceive()>(short)0) {
                    Util.arrayCopy(apdu.getBuffer(),(short)0x0005,menuTitle,(short)0x0000,
                                (short)0x0006);
                }
                break;
            default:
                ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);
         }
     }
}
```

Annex E (informative): Change history

The table below indicates all change requests that have been incorporated into the present document.

Change history								
Date	**TSG #**	**TSG Doc**	**CR**	**Rev**	**Cat**	**Subject/Comment**	**Old**	**New**
2001-06	TP-12	-	-			TS 43.019 version 4.0.0 created from TS 03.19 version 8.2.0. No technical changes were introduced		4.0.0
2001-12	TP-14	TP-010241	-		F	Clarification of ToolkitException. OUT_OF_TLV_BOUNDARIES in ViewHandler.java	4.0.0	4.1.0

History

Document history		
V4.0.0	June 2001	Publication
V4.1.0	December 2001	Publication