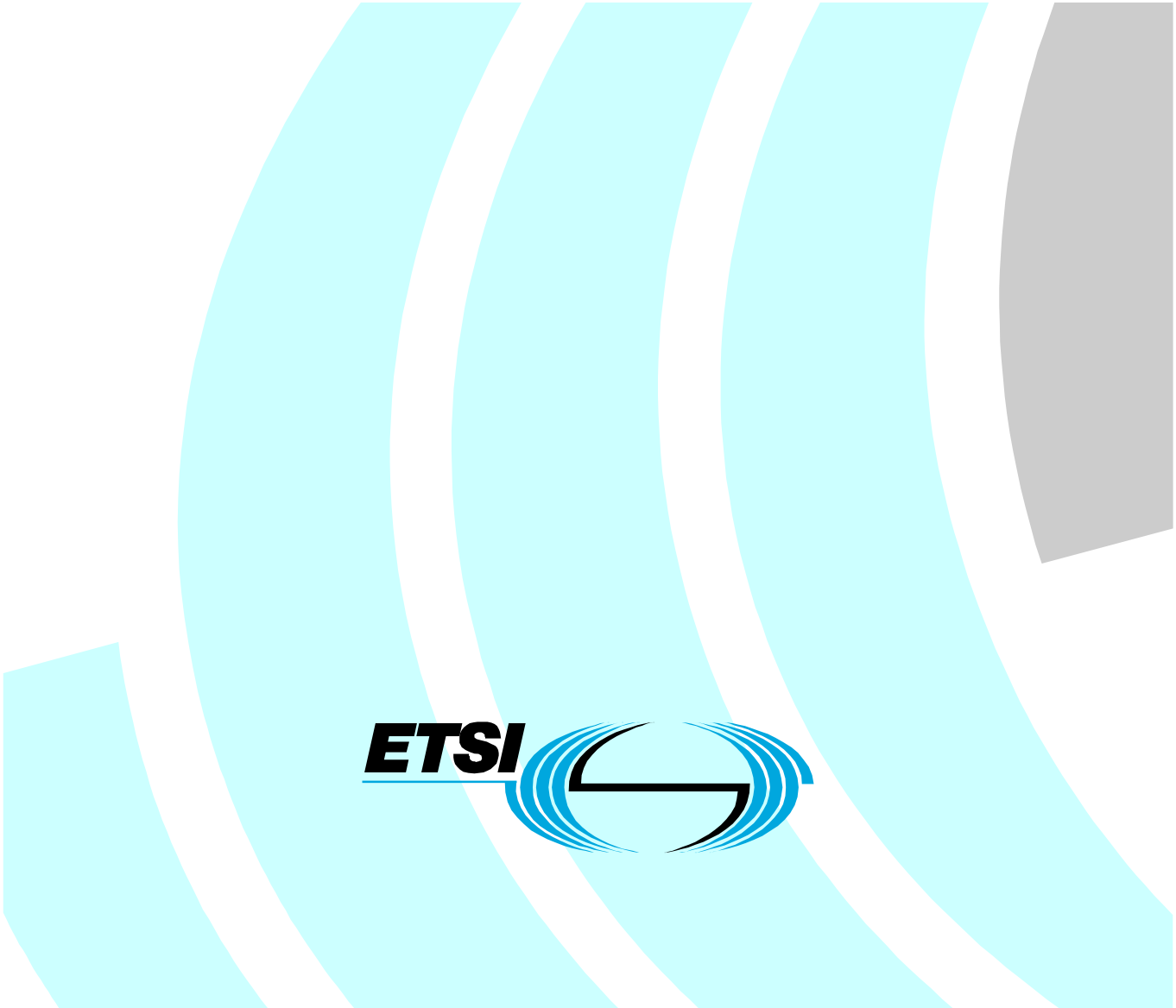
ETSI TS 102 241 V6.0.0 (2003-06)

Technical Specification

Smart Cards;
UICC Application Programming Interface (UICC API)
for Java Card™;
(Release 6)



Reference
DTS/SCP-030309

Keywords
API, smart card

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

Contents

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 Project Smart Card Platform (SCP).

The present document details the stage 1 aspects (overall service description) for the support of a "Application Programming Interface and Loader Requirements" [11].

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

Version 1.x.y, where:

- x the second digit is incremented for changes of substance, i.e. technical enhancements, corrections, updates, etc.
- y the third digit is incremented when editorial only changes have been incorporated in the specification;

1 Scope

The present document defines the stage two description of the "Application Programming Interface and Loader Requirements" [11] internal to the UICC.

This stage two describes the functional capabilities and the information flow for the UICC API implemented on the Java Card™ 2.2 specification [2], [3] and [4].

The present document includes information applicable to network operators, service providers and UICC, 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 and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

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

[2] Sun Microsystems Java Card™ Specification: "Java Card™ 2.2 Application Programming Interface".

[3] Sun Microsystems Java Card™ Specification: "Java Card™ 2.2 Runtime Environment (JCRE) Specification".

[4] Sun Microsystems Java Card™ Specification: "Java Card™ 2.2 Virtual Machine Specification".

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

[5] ETSI TS 101 220: "Smart Cards; ETSI numbering system for telecommunication application providers (Release 6)".

[6] ETSI TS 102 221: "Smart cards; UICC-Terminal interface; Physical and logical characteristics (Release 6)".

[7] ETSI TS 102 223: "Smart cards; Card Application Toolkit (CAT) (Release 6)".

[8] ETSI TS 102 222: "Integrated Circuit Cards (ICC); Administrative commands for telecommunications applications (Release 6)".

[9] ETSI TS 102 225: "Smart Cards; Secured packet structure for UICC based applications (Release 6)".

[10] ETSI TS 102 226: "Smart Cards; Remote APDU structure for UICC based applications (Release 6)".

[11] ETSI TS 102 240: "Smart Cards; UICC Application Programming Interface and Loader Requirements; Service description; (Release 6)".

[12] ETSI TS 151 011: "Digital cellular telecommunications system (Phase 2+); Specification of the Subscriber Identity Module - Mobile Equipment (SIM-ME) interface (3GPP TS 51.011 version 4.7.0 Release 4)".

3 Definitions and abbreviations

3.1 Definitions

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

applet : application built up using a number of classes which will run under the control of the Java Card virtual machine

bytecode : machine independent code generated by a Java compiler and executed by the Java interpreter

class : type that defines the implementation of a particular kind of object

NOTE: A Class definition defines instance and class variables and methods.

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

java : object oriented programming language developed by Sun Microsystems designed to be platform independent

method : piece of executable code that can be invoked, possibly passing it certain values as arguments

NOTE: Every Method definition belongs to some class.

object : principal building block of object oriented programs

NOTE: Each object is a programming unit consisting of data (variables) and functionality (methods).

package : group of classes

NOTE: Packages are declared when writing a Java Card program

toolkit application :application on the UICC card which can be triggered by toolkit events issued by the Terminal and which can send proactive commands to the terminal

NOTE: These applications can be downloaded via any type of network.

virtual machine : 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:

AID	Application Identifier
APDU	Application Protocol Data Unit
API	Application Programming Interface
FFS	For Further Study
JCRE	Java Card™ Run Time Environment
NAA	Network Access Application (e.g. SIM, USIM)

4 Description

The present document describes an API and a Runtime Environment for the UICC platform. This API and the Runtime Environment allows application programmers to get access to the functions and data described in TS 102 221 [6] and TS 102 223 [7] such that UICC based services can be developed and loaded onto a UICC, quickly and, if necessarily, remotely, after the card has been issued.

This API is an extension to the "Java Card™ 2.2 API" [2], the Runtime Environment is an extension of the "Java Card™ 2.2 Runtime Environment" [3].

4.1 UICC Java Card Architecture

The over all architecture of the UICC API is based on Java Card™ 2.2 [2], [3] and [4]:

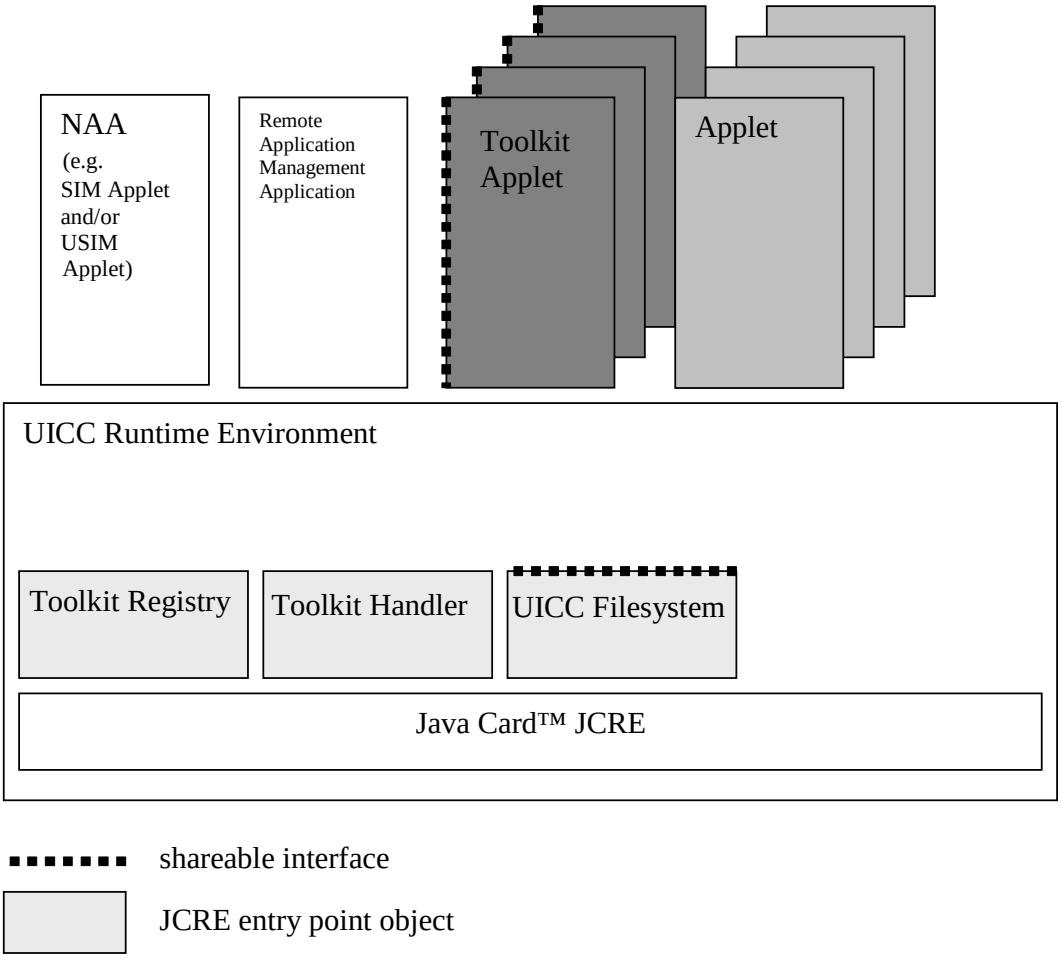


Figure 1: UICC Java Card™ Architecture

UICC Toolkit Framework: this is the UICC Java Card™ runtime environment, it is composed of the JCRE, the Toolkit Registry, the Toolkit Handler and the File Systems.

JCRE: this is specified in "Java Card™ 2.2 Runtime Environment (JCRE) Specification" [3] 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).

UICC File System: it contains the File System of the UICC specified in TS 102 221 [6] (i.e. the files under the MF level and the DF $_{Telecom}$) and handles its own file access control. It is a JCRE owned object implementing the shareable interface *uicc.access.UICCView*.

Applets: these derive from *javacard.framework.Applet* and provide the entry points : *process, select, deselect, install* as defined in the "Java Card™ 2.2 Runtime Environment Specification" [3].

Toolkit applets: are the Java Card based implementation of Toolkit Applications, these derive from *javacard.framework.Applet*, to provide the same entry points, and provide one object implementing the *uicc.toolkit.ToolkitInterface* shareable interface, so that these applets can be triggered by an invocation of the *processToolkit()* method. The Toolkit Applet(s) AID are defined in TS 101 220 [5]

Remote Application Management Application : this is handling the loading, installation, management and removal of applets and packages as specified in TS 102 226 [10].

Shareable interface: this is defined in the Java Card™ 2.2 specifications [2], [3] and [4].

CAT session: card session opened by a terminal supporting proactive UICC, starting with the download of the Terminal Profile and ending with a subsequent reset or deactivation of the card.

5 File Access API

The file access API consists of the *uicc.access* package, which allows applets to access the file systems of the UICC.

5.1 FileView

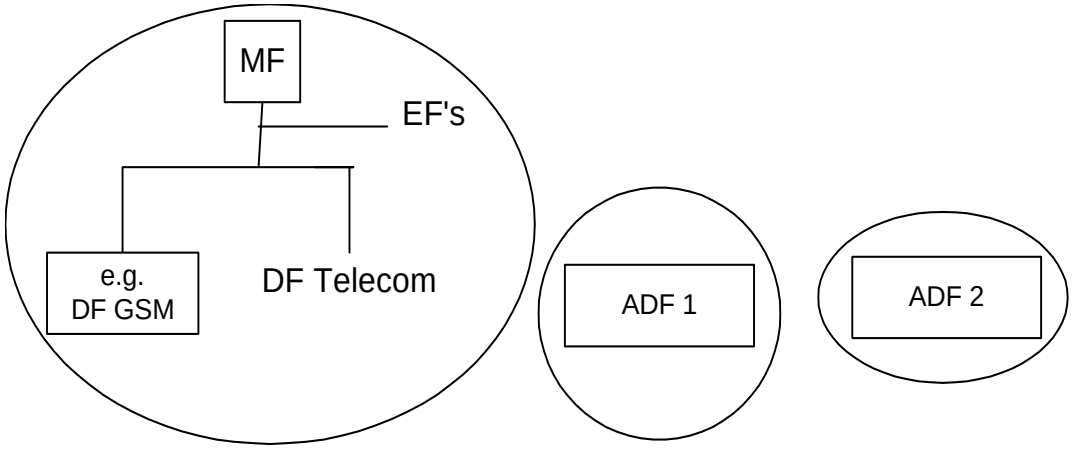


Figure 2: Logical structure of FileView

Any Applet (not only Toolkit Applets) is allowed to retrieve and use a *FileView*.

A *FileView* object can be retrieved by invoking one of the *getFileView()* methods defined in the *UICCSystem* class.

The UICC *FileView* allows to access the MF and all DFs and EFs that are located under the MF, including DF Telecom and any access technology specific DF located under the MF, but not the files located under any ADF. This *FileView* can be retrieved by invoking the *getFileView()* method from the *UICCSystem*. The only way to access the DF GSM is to request the UICC FileView.

An ADF *FileView* allows to access only the DFs and EFs located under the ADF. It is not possible to access the MF or any DF or EF located under the MF from an ADF *FileView*. An ADF *FileView* can be retrieved by invoking the *getFileView(…)* method with passing as parameter the full AID of the application owning the ADF.

Each *FileView* object shall be provided as a permanent JCRE entry point object.

A separate and independent file context shall be associated with each and every *FileView* object: the operation performed on files in a given *FileView* object shall not affect the file context associated with any other *FileView* object.

This context can be transient or persistent depending on what was required by the Applet during the creation of the *FileView* object.

Each *FileView* shall be given the access control privileges associated with the UICC or the corresponding ADF for the Applet. The access control privileges are checked each time a method of the *FileView* object is invoked. The access control privileges are defined by the access domain parameters specified in TS 102 226 [10].

The root of the context of a FileView object is the MF for the UICC FileView or the ADF for an ADF FileView.

At the creation of a *FileView* object, the current DF of the FileView's context is the root. When the transient context of a FileView is cleared, the current DF becomes the root of the FileView.

5.2 UICC file access

The following functions are provided by the methods defined in the *uicc.access.FileView* interface see annex A:

ACTIVATE FILE as defined in TS 102 222 [8]

DEACTIVATE FILE as defined in TS 102 222 [8]

INCREASE as defined in TS 102 221 [6]

READ BINARY as defined in TS 102 221 [6]

READ RECORD as defined in TS 102 221 [6]

SEARCH RECORD as defined in TS 102 221 [6]

SELECT by File ID or by Path as defined in TS 102 221 [6]

STATUS as defined in TS 102 221 [6]

UPDATE BINARY as defined in TS 102 221 [6]

UPDATE RECORD as defined in TS 102 221 [6]

6 UICC Toolkit Framework

The toolkit API is part of the "UICC Toolkit Framework" and consists of the *uicc.toolkit* package, which allows applets to access the toolkit features defined in TS 102 223 [7].

NOTE A new architectural drawing is FFS.

6.1 Applet Triggering

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

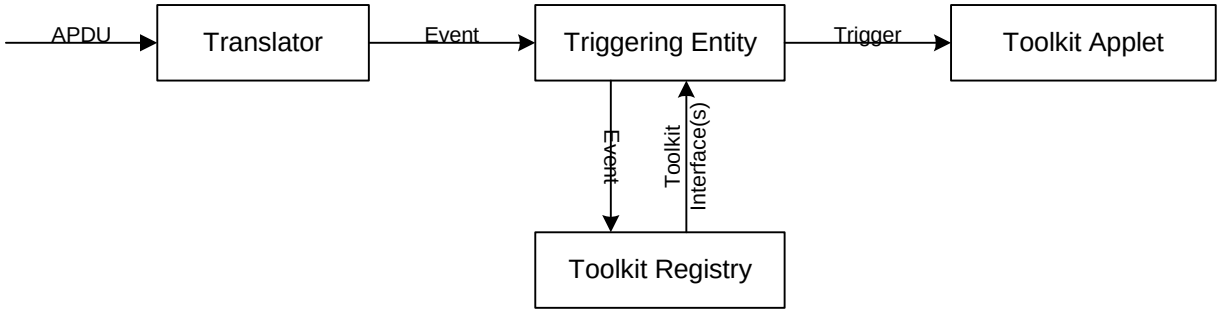


Figure 3: toolkit applet triggering diagram

The Translator converts the information from an incoming APDU into the corresponding Event information.

The Triggering Entity requests the information from the Toolkit Registry, which Toolkit Applets are registered to this Event. The Triggering Entity then triggers the Toolkit Applet. The terminal shall not be adversely affected by the presence of applets on the UICC card. For instance a syntactically correct Envelope shall not result in an error status word in case of a failure of an applet. The applications seen by the terminal are first level applications (e.g. SIM, USIM). A Toolkit Applet may throw an exception, but this error shall not be sent to the terminal.

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 written above, when a first level application 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 selected.

The UICC Toolkit Framework shall only trigger a Toolkit Applet if it is in the selectable state as defined in TS 102 226 [10].

The UICC Toolkit Framework shall trigger the Toolkit Applets according to their priority level assigned at installation time. The priority level specifies the order of activation of an applet compared to the other applets registered to the same event. If two or more applets are registered to the same event and have the same priority level, the applet are triggered according to their installation time (i.e. the most recent applet is activated first). TS 102 226 [10] defined the priority level coding and how this parameter is provided to the UICC.

When the UICC Toolkit Framework has to trigger several applets on the same event, the next applet is triggered on the return of the *processToolkit()* method of the previous Toolkit Applet.

6.2 Definition of Events

The following events can trigger a Toolkit Applet:

Table 1: UICC toolkit event list

Event Name	Reserved short value
not to be used	0
EVENT_PROFILE_DOWNLOAD	1
Reserved by 3GPP	2
Reserved by 3GPP	3
Reserved by 3GPP	4
Reserved by 3GPP	5
Reserved by 3GPP	6
EVENT_MENU_SELECTION	7
EVENT_MENU_SELECTION_HELP_REQUEST	8
EVENT_CALL_CONTROL_BY_NAA	9
Reserved by 3GPP	10
EVENT_TIMER_EXPIRATION	11
EVENT_EVENT_DOWNLOAD_MT_CALL	12
EVENT_EVENT_DOWNLOAD_CALL_CONNECTED	13
EVENT_EVENT_DOWNLOAD_CALL_DISCONNECTED	14
EVENT_EVENT_DOWNLOAD_LOCATION_STATUS	15
EVENT_EVENT_DOWNLOAD_USER_ACTIVITY	16
EVENT_EVENT_DOWNLOAD_IDLE_SCREEN_AVAILABLE	17
EVENT_EVENT_DOWNLOAD_CARD_READER_STATUS	18
EVENT_STATUS_COMMAND	19
EVENT_EVENT_DOWNLOAD_LANGUAGE_SELECTION	20
EVENT_EVENT_DOWNLOAD_BROWSER_TERMINATION	21
EVENT_EVENT_DOWNLOAD_DATA_AVAILABLE	22
EVENT_EVENT_DOWNLOAD_CHANNEL_STATUS	23
EVENT_FORMATTED_SMS_CB (reserved for GSM/3GPP)	24
EVENT_EVENT_ACCESS_TECHNOLOGY_CHANGE	25
EVENT_EVENT_DISPLAY_PARAMETER_CHANGE	26
EVENT_EVENT_LOCAL_CONNECTION	27
RFU	28 to 125
EVENT_APPLICATION_DESELECT	126
RFU	127
RFU	128 to 32767
EVENT_UNRECOGNIZED_ENVELOPE	-1
Reserved for Proprietary Use	-2 to (-32768)

EVENT_PROFILE_DOWNLOAD

Upon reception of a TERMINAL PROFILE APDU as defined in TS 102 221 [6] and TS 151 011 [12] command the UICC Toolkit Framework shall store the terminal profile and trigger all the Toolkit Applet(s) registered to this event.

EVENT_MENU_SELECTION, EVENT_MENU_SELECTION_HELP_REQUEST

Upon reception of an ENVELOPE (MENU SELECTION) APDU as defined in TS 102 221 [6] and TS 151 011 [12] command the UICC toolkit framework shall only trigger the Toolkit Applet registered to the corresponding event with the associated menu identifier.

A Toolkit Applet shall be triggered by the EVENT_MENU_SELECTION_HELP_REQUEST event only if help is available for the corresponding Menu entry.

EVENT_CALL_CONTROL_BY_NAA

Upon reception of an ENVELOPE (CALL CONTROL) APDU as defined in TS 102 221 [6] and TS 151 011 [12] the UICC Toolkit Framework shall trigger the Toolkit Applet registered to this event. Regardless of the Toolkit Applet state the UICC Toolkit Framework shall not allow more than one Toolkit Applet to be registered to this event at a time, in particular, if a Toolkit Applet is registered to this event but not in selectable state the UICC Toolkit Framework shall not allow another Toolkit Applet to register to this event.

EVENT_TIMER_EXPIRATION

Upon reception of an ENVELOPE (TIMER EXPIRATION) APDU as defined in TS 102 221 [6] and TS 151 011 [12] command the UICC Toolkit Framework shall only trigger the Toolkit Applet registered to this event with the associated timer identifier.

EVENT_EVENT_DOWNLOAD_MT_CALL,

EVENT_EVENT_DOWNLOAD_CALL_CONNECTED,

EVENT_EVENT_DOWNLOAD_CALL_DISCONNECTED,

EVENT_EVENT_DOWNLOAD_LOCATION_STATUS,

EVENT_EVENT_DOWNLOAD_USER_ACTIVITY,

EVENT_EVENT_DOWNLOAD_IDLE_SCREEN_AVAILABLE,

EVENT_EVENT_DOWNLOAD_CARD_READER_STATUS,

EVENT_EVENT_DOWNLOAD_LANGUAGE_SELECTION,

EVENT_EVENT_DOWNLOAD_BROWSER_TERMINATION,

EVENT_EVENT_DOWNLOAD_ACCESS_TECHNOLOGY_CHANGE,

EVENT_EVENT_DOWNLOAD_DISPLAY_PARAMETER_CHANGED,

EVENT_EVENT_DOWNLOAD_LOCAL_CONNECTION

Upon reception of an ENVELOPE (Event Download) APDU as defined in TS 102 221 [6] and TS 151 011 [12] command the UICC Toolkit Framework shall trigger all the Toolkit Applets registered to the corresponding event.

EVENT_EVENT_DOWNLOAD_DATA_AVAILABLE,

EVENT_EVENT_DOWNLOAD_CHANNEL_STATUS

Upon reception of an ENVELOPE (Event Download) APDU as defined in TS 102 221 [6] and TS 151 011 [12] command the UICC Toolkit Framework shall only trigger the Toolkit Applet registered to the corresponding event with the associated channel identifier.

The registration to these events is effective once the toolkit applet has issued a successful OPEN CHANNEL proactive command, and is valid until the first successful CLOSE CHANNEL with the corresponding channel identifier, or the end of the card session.

When a Toolkit Applet sends an OPEN CHANNEL proactive command and receives a TERMINAL RESPONSE with General Result = "0x0X", the framework shall assign the channel identifier to the calling Toolkit Applet.

When a Toolkit Applet sends a CLOSE CHANNEL proactive command and receives a TERMINAL RESPONSE with General Result ="0x0X", the framework shall release the corresponding channel identifier.

EVENT_STATUS_COMMAND

Upon reception of an STATUS APDU as defined in TS 102 221 [6] and TS 151 011 [12] command the UICC Toolkit Framework shall trigger all the Toolkit Applet(s) registered to this event.

EVENT_APPLICATION_DESELECT

When an application session is terminated (as described in TS 102 221 [6]) the UICC Toolkit Framework shall trigger all the Toolkit Applets registered to this event. The AID of the deselected application is available to the Toolkit Applet in the *EnvelopeHandler*, as an AID Simple TLV data object as defined in the TS 102 223 [7].

The *ProactiveHandler* is not available for triggered Toolkit Applets during the processing of this event.

EVENT_FIRST_COMMAND_AFTER_ATR

FFS: This section is for further study.

EVENT_UNRECOGNIZED_ENVELOPE

Upon reception of an unrecognized ENVELOPE APDU as defined in TS 102 221 [6] and TS 151 011 [12] command the UICC Toolkit Framework shall trigger all the Toolkit Applet(s) registered to this event.

An ENVELOPE APDU command shall be considered as unrecognized by the UICC Toolkit Framework if its BER-TLV tag is not defined in the *ToolkitConstants* interface. The EVENT_UNRECOGNIZED_ENVELOPE event allows a Toolkit Applet to handle the evolution of the TS 102 223 [7] specification.

As a consequence of the *EnvelopeResponseHandler* availability rules specified in clause 6.6, only the first triggered toolkit applet is guaranteed to be able to post a response.

6.3 Registration

A Toolkit Applet shall register to the JCRE as specified in "Java Card™ 2.2 Runtime Environment (JCRE) Specification" [3].

A Toolkit Applet shall register to the UICC Toolkit Framework, by calling the *ToolkitRegistrySystem.getEntry()* method. A Toolkit Applet can change its registration to toolkit events during its whole life cycle.

The registration of a Toolkit Applet to an event shall not be affected by its life cycle state, in particular a Toolkit Applet shall still be considered as registered to an event if it is not in the *selectable* life cycle state.

The toolkit events registration API is described in the *uicc.toolkit.ToolkitRegistry* interface in annex A.

6.4 Proactive command handling

The UICC Toolkit Framework is in charge of managing the toolkit protocol for the Toolkit Applet(s) (i.e. 91xx, Fetch, Terminal Response).

The *uicc.toolkit.ProactiveHandler* API defines the methods made available to Toolkit Applets by the UICC Toolkit Framework so that the Toolkit Applets can:

- initialize a proactive command with the *init()* method ;
- append several Simple TLV as defined in TS 102 223 [7] to the proactive command with the *appendTLV()* methods ;
- request the UICC Toolkit Framework to send this proactive command to the terminal and wait for the response, with the *send()* method.

On the call to the *send()* method the UICC Toolkit Framework shall handle the transmission of the proactive command to the terminal, and the reception of the response. On the return from the *send()* method the UICC Toolkit Framework shall resume the Toolkit Applet execution. . It shall provide to the Toolkit Applet the *uicc.toolkit.ProactiveResponseHandler*, so that the toolkit applet can analyse the response.

The UICC Toolkit Framework shall prevent the Toolkit Applet from sending the following system proactive commands: SET UP MENU, SET UP EVENT LIST, POLL INTERVAL, POLLING OFF. If an applet attempts to send such a command, the UICC Toolkit Framework shall throw an exception.

The UICC Toolkit Framework shall prevent a Toolkit Applet from sending a TIMER MANAGEMENT proactive command using a timer identifier, which is not allocated to it. If an applet attempts to send such a command, the UICC Toolkit Framework shall throw an exception.

The UICC Toolkit Framework shall prevent a Toolkit Applet from sending a SEND DATA, RECEIVE DATA and CLOSE CHANNEL proactive commands using a channel identifier, which is not allocated to it. If an applet attempts to send such a command the UICC Toolkit Framework shall throw an exception.

The UICC Toolkit Framework shall prevent a Toolkit Applet from sending an OPEN CHANNEL proactive command if it exceeds the maximum number of channel allocated to this applet. If an applet attempts to send such a command the UICC Toolkit Framework shall throw an exception.

All other proactive commands shall be sent to the terminal as constructed by the Toolkit Applet without any check by the UICC Toolkit Framework.

The UICC 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

The *uicc.toolkit.EnvelopeResponseHandler* API defines the methods made available to Toolkit Applets by the UICC Toolkit Framework so that the Toolkit Applets can send a response to some specific events.
(e.g. EVENT_CALL_CONTROL)

A Toolkit Applet can post a response to some events with the *post()* or the *postAsBERTLV()* methods and can continue its processing after the call to these methods.

The UICC Toolkit Framework shall send the response before the emission of the next proactive command or when all the Toolkit Applets triggered by the event have finished their processing.

6.6 System Handler management

The system handlers: *ProactiveHandler*, *ProactiveResponseHandler*, *EnvelopeHandler* and *EnvelopeResponseHandler* are Temporary JCRE Entry Point Object as defined in the "Java Card™ 2.2 Runtime Environment (JCRE) Specification" [3].

A system handler is considered available when no HANDLER_NOT_AVAILABLE *ToolkitException* is thrown when the corresponding *getTheHandler()* method is called or a method of the handler is called.

The availability and the content of the system handlers are not defined outside the applet triggering as defined in the present document.

The following rules define the availability and the content of the system handlers. These are generic rules and may vary with the event that triggers the toolkit applet.

ProactiveHandler:

- The *ProactiveHandler* shall not be available if the Terminal Profile command has not yet been processed by the UICC Toolkit Framework.
- When available the *ProactiveHandler* shall remain available until 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 its init method the content is cleared and then initialized.
- After a call to *ProactiveHandler.send()* method the content of the handler shall not be modified by the UICC Toolkit Framework.

ProactiveResponseHandler:

- The *ProactiveResponseHandler* shall be available as soon as the *ProactiveHandler* is available, its TLV list shall be empty before the first call to the *ProactiveHandler.send()* method. Shall remain available until the termination of the *processToolkit()* method.
- The *ProactiveResponseHandler* shall not be available if the *ProactiveHandler* is not available.
- The *ProactiveResponseHandler* TLV list is filled with the simple TLV data objects of the last TERMINAL RESPONSE APDU command. The simple TLV data objects shall be provided in the order given in the TERMINAL RESPONSE command data.
- The *ProactiveResponseHandler* content shall be updated after each successful call to *ProactiveHandler.send()* method and shall remains unchanged until the next successful call to the *ProactiveHandler.send()* method.

EnvelopeHandler:

- When available (as specified in table 1) the *EnvelopeHandler* shall remain available and its content shall remain unchanged from the invocation to the termination of the *processToolkit()* method.
- The *EnvelopeHandler* TLV list is filled with the simple TLV data objects of the ENVELOPE APDU command. The simple TLV data objects shall be provided in the order given in the ENVELOPE command data.

EnvelopeResponseHandler:

- The *EnvelopeResponseHandler* is available (as specified in Table 1) 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.
- After the first invocation of the *ProactiveHandler.send()* method the *EnvelopeResponseHandler* is no more available.
- At the *processToolkit()* method invocation the TLV-List is cleared.

The table 2 describes the minimum availability of the handlers for all the events at the invocation of the *processToolkit()* method of the Toolkit Applet.

Table 2: Handler availability for each event

EVENT_	Reply busy allowed	Envelope Handler	Envelope ResponseHandler	Nb of triggered/ registered Applet
_MENU_SELECTION	Y	Y	N	1/n (per Item Id)
_MENU_SELECTION_HELP_REQUEST	Y	Y	N	1/n (per Item Id)
_CALL_CONTROL_BY_NAA	N	Y	Y	1/1
_TIMER_EXPIRATION	Y	Y	N	1/8 (per timer) (see note)
_EVENT_DOWNLOAD				
_MT_CALL	Y	Y	N	n/n
_CALL_CONNECTED	Y	Y	N	n/n
_CALL_DISCONNECTED	Y	Y	N	n/n
_LOCATION_STATUS	Y	Y	N	n/n
_USER_ACTIVITY	Y	Y	N	n/n
_IDLE_SCREEN_AVAILABLE	Y	Y	N	n/n
_CARD_READER_STATUS	Y	Y	N	n/n
_LANGUAGE_SELECTION	Y	Y	N	n/n
_BROWSER_TERMINATION	Y	Y	N	n/n
_DATA_AVAILABLE	Y	Y	N	1/7 (per channel) (see note)
_CHANNEL_STATUS	Y	Y	N	1/7 (per channel) (see note)
__ACCESS_TECHNOLOGY_CHANGE	Y	Y	N	n/n
__DISPLAY_PARAMETER_CHANGE	Y	Y	N	n/n
__LOCAL_CONNECTION	Y	Y	N	n/n
_UNRECOGNIZED_ENVELOPE	Y	Y	Y	n/n
_STATUS_COMMAND	N	N	N	n/n
_PROFILE_DOWNLOAD	N	N	N	n/n
NOTE: One toolkit applet can register to several timers/channels, but a timer/channel can only be allocated to one toolkit applet.				

6.7 UICC Toolkit Framework behaviour

The following rules define the UICC Toolkit Framework behaviour for:

- ToolkitInterface shareable interface object retrieval:
  - The UICC Toolkit Framework shall invoke the *getShareableInterfaceObject()* method of the Toolkit Applet to retrieve the reference to its shareable *ToolkitInterface* object, before triggering it the first time in its life cycle.
  - The byte parameter of the *getShareableInterfaceObject()* method shall be set to one (i.e. "01").
  - FFS: The AID parameter of the *getShareableInterfaceObject()* method shall be set to the AID of the UICC Toolkit Framework or to null.
- Triggering of a Toolkit Applet (invocation of the *processToolkit()* method from the *ToolkitInterface* shareable interface):
  - The UICC Toolkit Framework triggers a Toolkit Applet by calling the *processToolkit()* method of the *ToolkitInterface* shareable interface object provided by the Toolkit Applet. As a consequence all the rules defined in "Java Card™ 2.2 (JCRE) Runtime Environment Specification" [3] apply (e.g. access to CLEAR_ON_DESELECT transient objects, context switch, multi selectable).
  - At the invocation of the *processToolkit()* method there shall be no transaction in progress.
- Termination of a Toolkit Applet (return from the *processToolkit()* method):
  - A pending toolkit applet transaction is aborted.
- 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.
  - A pending toolkit applet transaction at the method invocation is aborted.

6.7.1 System Proactive Commands

The system proactive command shall only contain information from Toolkit Applets that are in the selectable state.

The UICC Toolkit Framework shall send its system proactive command(s) as soon as no proactive session is ongoing and after all the Toolkit Applets registered to the current events have been triggered and have returned from the *processToolkit()* method invocation.

6.7.1.1 SET UP MENU

At the beginning of a CAT session, the UICC Toolkit Framework shall send a SET UP MENU system proactive command, if at least one menu entry is registered and enabled by a selectable Toolkit Applet.

During a CAT session the UICC Toolkit Framework shall send a SET UP MENU system proactive command whenever a menu entry is modified, added or removed.

If help is available for at least one Menu Entry inserted in the SET UP MENU system proactive command the UICC Toolkit Framework shall indicate to the terminal that help information is available.

If help is not available for all Menu Entries inserted in the SET UP MENU system proactive command the UICC Toolkit Framework shall not indicate to the terminal that help information is available.

The UICC Toolkit Framework shall use the data of the EF_{sume} file under the DF_Telecom 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 provided at the installation of the toolkit applet.

- Item identifiers: The Item identifiers used in Item comprehension TLV of the SET UP MENU system proactive command are the ones returned by the *initMenuEntry(…)* method. The Item identifier values are split in two ranges.
  - The range (1,127) of the item identifier is managed by the Remote Application Management Application (TS 102 226 [10]) and provided to the UICC Toolkit Framework.
  - The range (128,255) is managed by the UICC Toolkit Framework. When the requested item identifier is "00" the UICC Toolkit Framework shall assign the first free value in the range (128,255).
- Item position: FFS

6.7.1.2 SET UP EVENT LIST

At the beginning of a CAT session, the UICC Toolkit Framework shall send a SET UP EVENT LIST system proactive command, if at least one event is registered by a selectable Toolkit Applet.

During a CAT session the UICC Toolkit Framework shall send a SET UP EVENT LIST system proactive command whenever the registered event list is changed.

6.7.1.3 POLL INTERVAL and POLLING OFF

At the beginning of a CAT session, the UICC Toolkit Framework shall send a POLL INTERVAL system proactive command, if at least one Toolkit Applet has requested a poll interval duration.

During a CAT session the UICC Toolkit Framework shall send a POLL INTERVAL or POLLING OFF system proactive command whenever the system poll interval duration is changed.

7 UICC toolkit applet

7.1 Applet Loading

The UICC API card shall be compliant to the "Java Card™ 2.2 Virtual Machine Specification" [4] and to annex B to guarantee interoperability at byte code Level.

The applet loading mechanism and applet life cycle are defined in TS 102 226 [10]. The applet loading protocol is defined in TS 102 225 [9].

7.2 Data and Function Sharing

The sharing mechanism defined in "Java Card™ 2.2 Application Programming Interface Specification" [2] and "Java Card™ 2.2 Runtime Environment Specification" [3] shall be used by the Toolkit Applet(s) to share data and function.

7.3 Package, Applet and Object Deletion

The Package and Applet deletion mechanism defined in "Java Card™ 2.2 Runtime Environment Specification" [3] shall be used to delete the content from the UICC. The object deletion mechanism defined optional in "Java Card™ 2.2 Application Programming Interface Specification" [2] is FFS (i.e.: mandatory or optional).

If an object deletion mechanism is supported then the one defined in "Java Card™ 2.2 Application Programming Interface Specification" [2] shall be used.

NOTE: The maximum work waiting time depends on several factors (e.g. the permissible duration of a network-UICC authentication); in some cases as little as 2 seconds could be required. During this period the UICC should respect the work waiting time procedure, defined in TS 102 221 [6] and TS 151 011 [12].

Annex A (normative): Java Card UICC API

The source files for the Java Card UICC API (Annex_A_Java.zip and Annex_A_HTML.zip) are contained in ts_102241v060000p0.zip, which accompanies the present document.

Annex B (normative): Java Card UICC API identifiers

The export files for the uicc.* package (Annex_B_Export_files.zip) are contained in ts_102241v060000p0.zip, which accompanies the present document.

NOTE: see the "Java Card™ 2.2 Virtual Machine Specification" [4].

Annex C (normative): UICC API package version management

The table C.1describes the relationship between each TS 102 241 specification version and its UICC API packages AID and Major, Minor versions defined in the export files.

Table C.1

TS 102 241	uicc.access package		uicc.toolkit package	
	AID	Major, Minor	AID	Major, Minor
	A0 00 00 00 09 00 05 FF FF FF FF 89 11 00 00 00	1.0	A0 00 00 00 09 00 05 FF FF FF FF 89 12 00 00 00	1.0

The package AID coding is defined in TS 101 220 [5]. The UICC 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.

Change history

This annex lists all change requests approved for the present document since the first version was approved.

Meeting	Plenary Tdoc	WG tdoc	VERS	CR	REV	REL	CAT	SUBJECT	Resulting Version

History

Document history		
V6.0.0	June 2003	Publication