ETSI TS 102 241 V14.1.0 (2019-03)

Smart Cards;
UICC Application Programming Interface (UICC API)
for Java Card™
(Release 14)

Reference
RTS/SCP-T0310ve10

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

Essential patents

IPRs essential or potentially essential to normative deliverables 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 (https://ipr.etsi.org/).

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.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Smart Card Platform (SCP).

The present document details the stage 2 aspects (overall service description) for the support of an "Application Programming Interface and Loader Requirements" [11].

The contents of the present document are subject to continuing work within TC SCP and may change following formal TC SCP approval. If TC SCP decides to modify the contents of the present document, it will be re-released by TC SCP 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 TC SCP for information;

2 presented to TC SCP for approval;

3 or greater indicates TC SCP 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.

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

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™ Platform, 3.0.1 Classic Edition [2], [3] and [4].

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

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

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

- In the case of a reference to a TC SCP document, a non-specific reference implicitly refers to the latest version of that document in the same Release as the present document.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

[1] Void.

[2] ORACLE: "Application Programming Interface, Java Card™ Platform, 3.0.1 Classic Edition".

[3] ORACLE: "Runtime Environment Specification, Java Card™ Platform, 3.0.1 Classic Edition".

[4] ORACLE: "Virtual Machine Specification Java Card™ Platform, 3.0.1 Classic Edition".

NOTE: ORACLE Java Card™ Specifications can be downloaded at http://docs.oracle.com/javame/javacard/javacard.html.

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

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

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

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

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

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

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

[12] ETSI TS 123 040 (V6.6.0): "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); Technical realization of Short Message Service (SMS) (3GPP TS 23.040 version 6.6.0 Release 6)".

[13] ETSI TS 102 241: "Smart Cards; UICC Application Programming Interface (UICC API) for Java Card™".

[14] ETSI TS 102 671: "Smart Cards; Machine to Machine UICC; Physical and logical characteristics".

[15] GlobalPlatform: "Card Specification, version 2.3.1".

NOTE: See http://www.globalplatform.org/.

[16] GlobalPlatform: "Java Card API and Export File for Card Specification, v2.2.1", (org.globalplatform) v1.6.

NOTE: See http://www.globalplatform.org/.

[17] ETSI TS 102 613: "Smart Cards; UICC - Contactless Front-end (CLF) Interface; Physical and data link layer characteristics".

[18] ETSI TS 102 705: "Smart Cards; UICC Application Programming Interface for Java Card™ for Contactless Applications".

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

- In the case of a reference to a TC SCP document, a non-specific reference implicitly refers to the latest version of that document in the same Release as the present document.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

Not applicable.

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the following terms 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.

UICC suspended context: internal status of the UICC stored during a successful UICC suspension procedure according to ETSI TS 102 221 [6]

virtual machine: part of the Run-time environment responsible for interpreting the bytecode

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI TS 102 221 [6] and the following apply:

ADF	Application Dedicated File
AID	Application IDentifier
APDU	Application Protocol Data Unit
API	Application Programming Interface
DF	Dedicated File

NOTE: Abbreviation formerly used for Data Field.

EF	Elementary File
FFS	For Further Study
JCRE	Java Card™ Runtime Environment
MF	Master File
NAA	Network Access Application (e.g. SIM, USIM)
RFM	Remote File Management
TLV	Tag Length Value

4 Description

4.0 Purpose

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 ETSI TS 102 221 [6] and ETSI 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 "Application Programming Interface, Java Card™ Platform, 3.0.1 Classic Edition" [2], the Runtime Environment is an extension of the "Runtime Environment Specification, Java Card™ Platform, 3.0.1 Classic Edition" [3].

4.1 UICC Java Card™ architecture

The overall architecture of the UICC API is based on Java Card™ Platform, 3.0.1 Classic Edition [2], [3] and [4].

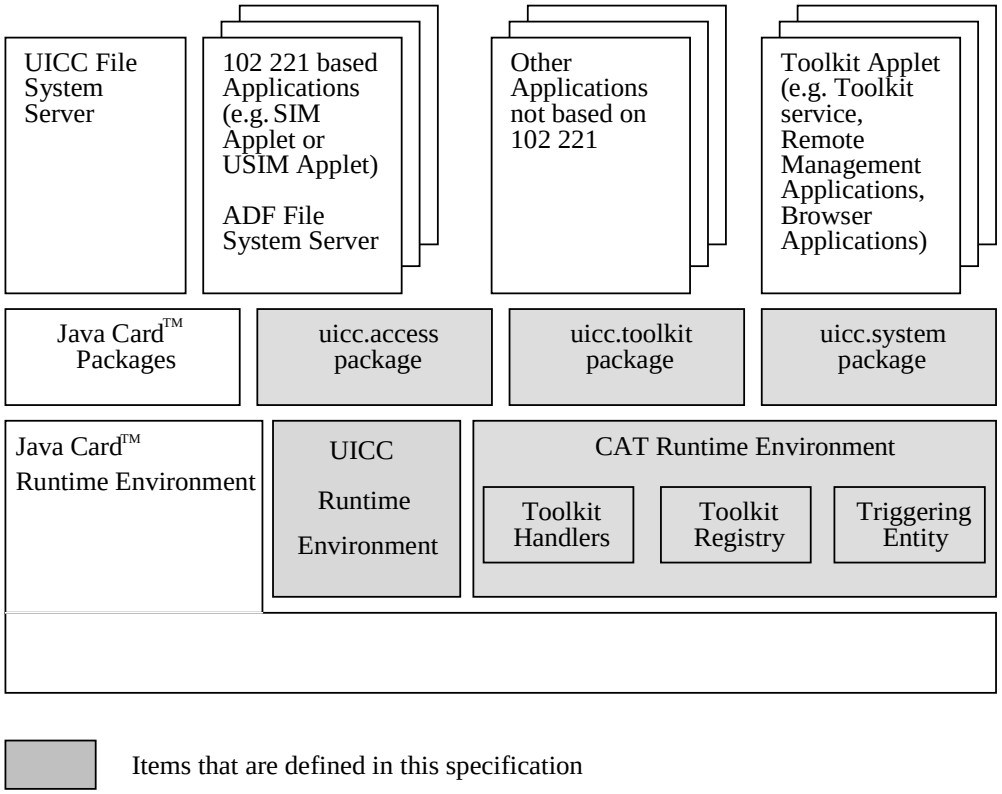

Figure 1: UICC Java Card™ architecture

Java Card™ Runtime Environment: this is specified in "Runtime Environment Specification, Java Card™ Platform, 3.0.1 Classic Edition" [3] and is able to select any specific applet and transmit to it the process of its APDU.

CAT Runtime Environment: this is the CAT Runtime Environment composed of, the Toolkit Registry, the Toolkit Handlers and the Triggering Entity. It is an addition to the JCRE.

UICC Runtime Environment: addition to the Java Card™ Runtime Environment.

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

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

UICC File System Server: it contains the File System of the UICC specified in ETSI TS 102 221 [6] (i.e. the EF and DF under the MF).

ADF File System Server: it contains the files of an ADF as specified in ETSI TS 102 221 [6] (i.e. the EF and DF under the ADF).

Applets: these derive from *javacard.framework.applet* and provide the entry points: *process, select, deselect, install* as defined in the " Runtime Environment Specification, Java Card™ Platform, 3.0.1 Classic Edition" [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* interface, so that these applets can be triggered by an invocation of the *processToolkit()* method. The Toolkit applet(s) AID are defined in ETSI TS 101 220 [5].

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

Shareable interface: this is defined in the "Runtime Environment Specification, Java Card™ Platform, 3.0.1 Classic Edition" 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

5.0 Introduction

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

5.1 FileView objects

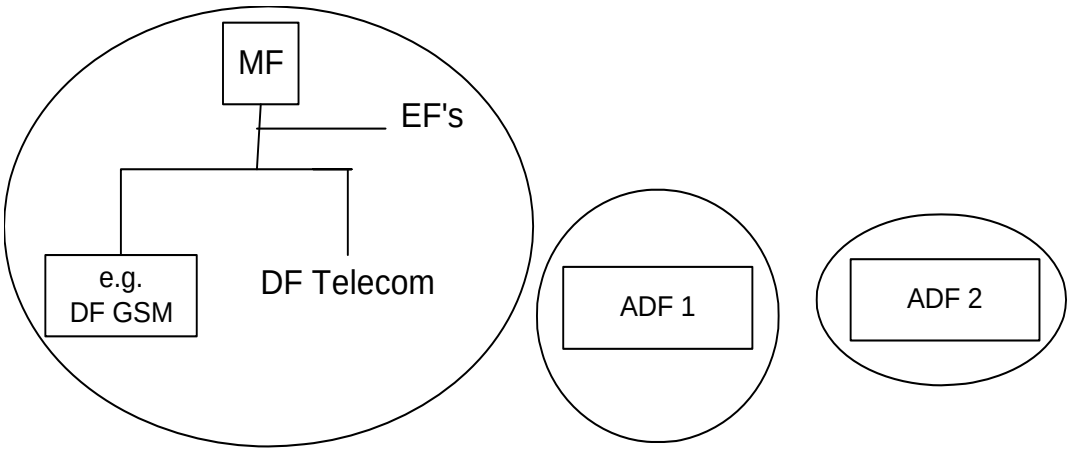

Figure 2: Logical structure of FileView

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 *getTheFileView()* 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 *getTheFileView()* 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 *getTheFileView(…)* 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 defined by the UICC access application specific parameters specified in ETSI TS 102 226 [10]. UICC administrative access application specific parameters shall not apply to objects retrieved from the *uicc.access.UICCSystem* class. The access control privileges are verified against the access rules defined in ETSI TS 102 221 [6] each time a method of the *FileView* object is invoked.

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 FileView operations

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

- ACTIVATE FILE as defined in ETSI TS 102 222 [8].
- DEACTIVATE FILE as defined in ETSI TS 102 222 [8].
- INCREASE as defined in ETSI TS 102 221 [6].
- READ BINARY as defined in ETSI TS 102 221 [6].
- READ RECORD as defined in ETSI TS 102 221 [6].
- SEARCH RECORD as defined in ETSI TS 102 221 [6].
- SELECT by File ID or by Path as defined in ETSI TS 102 221 [6].
- STATUS as defined in ETSI TS 102 221 [6].
- UPDATE BINARY as defined in ETSI TS 102 221 [6].
- UPDATE RECORD as defined in ETSI TS 102 221 [6].

5.3 BERTLVFileView operations

BER TLV files functions may be optionally supported by an implementation. If supported, an implementation shall provide the *uicc.access.bertlvfile* package and the 32-bit integer data type support defined optional in "Virtual Machine Specification Java Card™ Platform, 3.0.1 Classic Edition" [4] is mandatory.

The interface *uicc.access.bertlvfile.BERTLVFileView* extends the interface *uicc.access.FileView*, i.e. objects implementing the interface *BERTLVFileView* inherit *FileView* functionality.

If BER TLV files functions are supported by an implementation, the *getTheFileView()* and *getTheUICCView()* methods defined in the *UICCSystem* class shall return the reference of an object implementing the *BERTLVFileView* interface.

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

- RETRIEVE DATA as defined in ETSI TS 102 221 [6].
- SET DATA as defined in ETSI TS 102 221 [6].

6 Toolkit API and CAT Runtime Environment

6.0 Introduction

The toolkit API consists of the *uicc.toolkit* package, which allows applets to access the toolkit features defined in ETSI TS 102 223 [7].

6.1 Applet triggering

6.1.0 Triggering mechanism

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

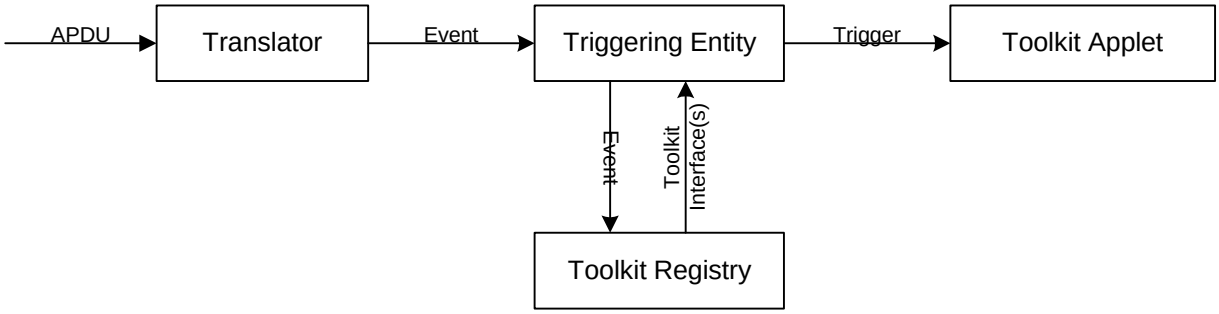

Figure 3: Toolkit applet triggering diagram

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).

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 CAT Runtime Environment shall only trigger a Toolkit applet if it is in the selectable state as defined in ETSI TS 102 226 [10].

The CAT Runtime Environment 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, except for the internal event EVENT_PROACTIVE_HANDLER_AVAILABLE (see clause 6.2), the applets are triggered according to their installation time (i.e. the most recent applet is activated first). ETSI TS 102 226 [10] defined the priority level coding and how this parameter is provided to the UICC.

When the CAT Runtime Environment 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.

If a UICC suspended context exists at the initiation of the card session (see clause 10), the CAT Runtime Environment shall not trigger applets on events (e.g. EVENT_FIRST_COMMAND_AFTER_ATR) but shall queue them. If the resume operation is successfully processed, this list of queued events shall be voided. Otherwise if the resume operation is cancelled (e.g. disallowed APDU command, bad resume token, etc.), the CAT Runtime Environment shall trigger Toolkit applets on queued events in the order of appearance of those events.

NOTE: When the resume operation is rejected, this is equivalent to a power off for applets selected at the time of the suspend operation as they are neither called on their *deselect()* method nor informed on the cancelled resume.

6.1.1 Exception handling

A Toolkit applet may throw an exception or an exception can occur during its processing. The CAT Runtime Environment shall catch any exception type or class and process as described here after.

If more than one applet shall be triggered by the currently processed event all Exceptions shall be caught by the CAT Runtime Environment and shall not be sent to the terminal. The CAT Runtime Environment shall proceed with the triggering.

If only one applet shall be triggered by the currently processed event and an ISOException with the following reason code is thrown it shall be sent to the terminal:

- ISOException with reason code REPLY_BUSY (0x9300).

Other Exceptions shall not be propagated to the terminal, this behaviour may be extended by an access technology depended specification.

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
Reserved by 3GPP	24
EVENT_EVENT_DOWNLOAD_ACCESS_TECHNOLOGY_CHANGE	25
EVENT_EVENT_DOWNLOAD_DISPLAY_PARAMETER_CHANGED	26
EVENT_EVENT_DOWNLOAD_LOCAL_CONNECTION	27
EVENT_EVENT_DOWNLOAD_NETWORK_SEARCH_MODE_CHANGE	28
EVENT_EVENT_DOWNLOAD_BROWSING_STATUS	29
Reserved by 3GPP	30
Reserved by 3GPP	31
EVENT_EVENT_DOWNLOAD_HCI_CONNECTIVITY	32
Reserved by 3GPP	33
EVENT_EVENT_DOWNLOAD_FRAMES_INFORMATION_CHANGED	34
EVENT_EVENT_DOWNLOAD_CONTACTLESS_STATE_REQUEST	35
EVENT_EVENT_POLL_INTERVAL_NEGOTIATION	36
RFU	37 to 118
Reserved by 3GPP	119
Reserved by 3GPP	120
Reserved by 3GPP	121
Reserved by 3GPP	122
EVENT_PROACTIVE_HANDLER_AVAILABLE	123
EVENT_EXTERNAL_FILE_UPDATE	124

Event Name	Reserved short value
EVENT_REMOTE_FILE_UPDATE	125
EVENT_APPLICATION_DESELECT	126
EVENT_FIRST_COMMAND_AFTER_ATR	127
EVENT_EVENT_DOWNLOAD_ACCESS_TECHNOLOGY_CHANGE_ MULTIPLE	128
EVENT_MEMORY_FAILURE	129
EVENT_TERMINAL_APPLICATIONS	130
RFU	131 to 32 767
EVENT_UNRECOGNIZED_ENVELOPE	-1
Reserved for Proprietary Use: - range for Card manufacturer proprietary events - range for Card Issuer proprietary events	- 2 to -64 -65 to -128
RFU	-129 to -32 768

EVENT_PROFILE_DOWNLOAD

Upon reception of a TERMINAL PROFILE APDU command as defined in ETSI TS 102 221 [6] the CAT Runtime Environment 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 command as defined in ETSI TS 102 221 [6] the CAT Runtime Environment 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 command as defined in ETSI TS 102 221 [6] the CAT Runtime Environment shall trigger the Toolkit applet registered to this event. Regardless of the Toolkit applet state the CAT Runtime Environment 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 CAT Runtime Environment shall not allow another Toolkit applet to register to this event.

EVENT_TIMER_EXPIRATION

Upon reception of an ENVELOPE (TIMER EXPIRATION) APDU command as defined in ETSI TS 102 221 [6] the CAT Runtime Environment 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_DISPLAY_PARAMETER_CHANGED

EVENT_EVENT_DOWNLOAD_NETWORK_SEARCH_MODE_CHANGE

EVENT_EVENT_DOWNLOAD_BROWSING_STATUS

EVENT_EVENT_DOWNLOAD_FRAMES_INFORMATION_CHANGED

EVENT_EVENT_DOWNLOAD_HCI_CONNECTIVITY

EVENT_EVENT_DOWNLOAD_CONTACTLESS_STATE_REQUEST

Upon reception of an ENVELOPE (Event Download) APDU command as defined in ETSI TS 102 221 [6] the CAT Runtime Environment shall trigger all the Toolkit applets registered to the corresponding event.

EVENT_EVENT_DOWNLOAD_ACCESS_TECHNOLOGY_CHANGE

Upon reception of an ENVELOPE (Event Download - Access Technology Change (single access technology)) APDU command as defined in ETSI TS 102 221 [6] the CAT Runtime Environment shall trigger all Toolkit applets registered to this event.

EVENT_EVENT_DOWNLOAD_ACCESS_TECHNOLOGY_CHANGE_MULTIPLE

Upon reception of an ENVELOPE (Event Download - Access Technology Change (multiple access technologies)) APDU command as defined in ETSI TS 102 221 [6] the CAT Runtime Environment shall trigger all Toolkit applets registered to this event.

EVENT_EVENT_DOWNLOAD_LOCAL_CONNECTION

Upon reception of an ENVELOPE (DOWNLOAD LOCAL CONNECTION) APDU as defined in ETSI TS 102 221 [6] command the CAT Runtime Environment shall only trigger the Toolkit applet registered to this event with the associated service identifier.

The registration to this event is effective once the Toolkit applet has issued a successful DECLARE SERVICE (add) proactive command, and is valid until the first successful DECLARE SERVICE (delete) with the corresponding service identifier, or the end of the card session.

EVENT_EVENT_DOWNLOAD_DATA_AVAILABLE

EVENT_EVENT_DOWNLOAD_CHANNEL_STATUS

Upon reception of an ENVELOPE (Event Download) APDU command as defined in ETSI TS 102 221 [6] the CAT Runtime Environment 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. It is valid to the end of the card session or to, the first successful CLOSE CHANNEL proactive command with the corresponding channel identifier.

A proactive command CLOSE CHANNEL for UICC Server Mode with command details set to "TCP in LISTEN state" does not affect the registration of the Toolkit applet to the event.

When a Toolkit applet sends an OPEN CHANNEL proactive command and receives a TERMINAL RESPONSE with General Result = "0x0X", the CAT Runtime Environment 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 CAT Runtime Environment shall release the corresponding channel identifier. An exception to this rule applies in the case of CLOSE CHANNEL for UICC Server Mode with command details set to "TCP in LISTEN state": When this proactive command is sent by a Toolkit applet and this applet receives a TERMINAL RESPONSE with General Result = "0x0X", the CAT Runtime Environment shall not release the corresponding channel identifier.

EVENT_STATUS_COMMAND

Upon reception of an STATUS APDU command as defined in ETSI TS 102 221 [6] the CAT Runtime Environment shall trigger all the Toolkit applet(s) registered to this event.

EVENT_APPLICATION_DESELECT

When an application session is terminated (as described in ETSI TS 102 221 [6]) the CAT Runtime Environment 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 ETSI TS 102 223 [7].

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

EVENT_FIRST_COMMAND_AFTER_ATR

Upon reception of the first APDU after either the ATR or the reception of the TERMINAL RESPONSE following the successful execution of a REFRESH with mode eUICC Profile State Change and before the Status Word related to this first APDU has been sent back by the UICC, the CAT Runtime Environment shall trigger all the Toolkit applet(s) registered to this event.

If the first APDU received is a Toolkit applet triggering APDU (e.g. TERMINAL PROFILE), the Toolkit applets registered to the EVENT_FIRST_COMMAND_AFTER_ATR event shall be triggered first.

The ProactiveHandler shall not be available at the invocation of the processToolkit method of the Toolkit applet on the EVENT_FIRST_COMMAND_AFTER_ATR event.

EVENT_UNRECOGNIZED_ENVELOPE

Upon reception of an unrecognized ENVELOPE APDU command as defined in ETSI TS 102 221 [6] the CAT Runtime Environment shall trigger all the Toolkit applet(s) registered to this event.

An ENVELOPE APDU command shall be considered as unrecognized by the CAT Runtime Environment if its BER-TLV tag is not defined in the *ToolkitConstants* interface or if the BER-TLV tag is reserved for GSM/3G/3GPP2 in ETSI TS 101 220 [5]. The EVENT_UNRECOGNIZED_ENVELOPE event allows a Toolkit applet to handle the evolution of the ETSI 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.

EVENT_PROACTIVE_HANDLER_AVAILABLE

The CAT Runtime Environment shall trigger all the Toolkit applets registered to this event when the *ProactiveHandler* is available and all the Toolkit applets registered to the previous event have been triggered and have returned from the *processToolkit()* invocation.

As with other events, the applet with the highest priority level and newest installation date shall be triggered first.

An applet that has the proactive handler may register for EVENT_PROACTIVE_HANDLER_AVAILABLE before returning to allow implementing a simple co-operative "task switching" mechanism based on priorities. Applets with the same priority level may implement "task switching" in a cyclic fashion.

If several applets have registered to EVENT_PROACTIVE_HANDLER_AVAILABLE and an applet returns from this event, the sequence of triggering shall be determined as follows:

- The list of registered applets shall be re-evaluated.
- If there is an applet with a higher priority level than the applet that returned, the applet with the highest priority shall be triggered.
- Else if there are one or more applet(s) with the same priority level as the applet that returned, all applets with this priority level shall be triggered in a cyclic fashion: As long as there is at least one applet with the same priority level and older installation date, the next older applet shall be triggered. If there is no older one, the applet with newest installation date shall be triggered.
- Else if there are only applet(s) left with lower priority level as the applet that returned, the applet with the next highest priority level and newest installation date shall be triggered.

When a Toolkit applet is triggered, it is automatically deregistered by the CAT Runtime Environment.

If the CAT session ends prior to an applet triggering, the applet will be triggered at the next CAT session.

NOTE 1: When the Toolkit applet is triggered the handlers' availability and content can be different from the content at the registration time. Therefore, the Toolkit applet has to store any handler data in order to use it in this event.

EVENT_EXTERNAL_FILE_UPDATE

Upon successful execution of an UPDATE BINARY or UPDATE RECORD or INCREASE or SET DATA APDU command (sent by the Terminal and received by the UICC on the I/O line) as defined in ETSI TS 102 221 [6], the CAT Runtime Environment shall trigger all the Toolkit applets registered to this event with the associated updated file. An applet shall only be triggered once per command.

Applet triggered upon execution of SET DATA command shall only occur once the related data object transfer is successfully completed.

When an applet is triggered by the *EVENT_EXTERNAL_FILE_UPDATE* event, the system EnvelopeHandler shall be made available, and shall contain the following COMPREHENSION TLVs (the order of the TLVs given in the system EnvelopeHandler is not specified):

- Device Identity with source set to terminal and destination set to UICC, as defined in ETSI TS 102 223 [7];
- File List, as defined in ETSI TS 102 223 [7]. The number of files shall be set to one. If a SFI referencing is used in the APDU Command, it shall be converted to its File Identifier;
- AID of the ADF, as defined in ETSI TS 102 223 [7], if the updated file belongs to an ADF. In this case, the path "3F007FFF" given in the File List indicates the ADF of the UICC application given through the AID. If the updated file belongs to the UICC shared file system, the AID TLV object is not present;
- File Update Information object.

  - In case of transparent file or record file:

Byte(s)	Description	Length
1	File Update Information tag	1
2	Length = 4	1
3 to 4	Position	2
5 to 6	Number of bytes updated	2

  Position depends on the file type:

  - In case of transparent file, Position = Offset;
  - In case of record file, Position = Absolute Record number.

  For the INCREASE APDU, the number of bytes updated is the record length.

  - In case of BER-TLV file, if a data object has been successfully updated:

Byte(s)	Description	Length
1	File Update Information tag	1
2	Length = T	1
3 to T+2	BER-TLV Tag of the updated data object	1 ≤ T ≤ 3
T+3	File Update Information tag	1
T+4	Length = L	1
T+5 to T+L+4	Length of the BER-TLV Value of the updated data object	1 ≤ L ≤ 4

  - In case of BER-TLV file, if a data object has been deleted or if a data object transfer has been aborted:

Byte(s)	Description	Length
1	File Update Information tag	1
2	Length = T	1
3 to T+2	BER-TLV Tag of the deleted data object	1 ≤ T ≤ 3

If a data object transfer has been aborted due to power loss, the event shall be generated at next card session.

The value returned upon a *getBERTag()* method invocation shall be equal to the BER-TLV tag for intra-UICC communication, as defined in ETSI TS 101 220 [5].

The registration to this event is effective once the applet has successfully called any of the methods registerFileEvent(…).

The deregistration for a particular file to this event is effective once the applet has successfully called any of the method deregisterFileEvent(…) whatever the method used to register was. A call to the method *clearEvent*(EVENT_EXTERNAL_FILE_UPDATE) clears the event EVENT_EXTERNAL_FILE_UPDATE from the Toolkit Registry of the applet. For all registered files, i.e. the applet is no longer triggered when a file which was previously registered is updated.

EVENT_REMOTE_FILE_UPDATE

This event shall be triggered on successful execution of a Remote File Management (RFM) command string containing one or several UPDATE BINARY or UPDATE RECORD or INCREASE APDU commands as defined in ETSI TS 102 226 [10] according to the following rules:

- The execution of the RFM command string shall be considered successful if at least one of the commands UPDATE BINARY, UPDATE RECORD or INCREASE APDU which are contained in it were successfully executed.
- The CAT Runtime Environment shall trigger all Toolkit applets registered to this event if at least one of the files contained in their list of registered files was updated. An applet which is not registered to any of the updated files shall not be triggered.
- An applet shall only be triggered once per RFM command string.
- Data provided in the system EnvelopeHandler shall not contain update information referring to the execution of more than one RFM command string.
- When an applet is triggered by the event *EVENT_REMOTE_FILE_UPDATE* the system EnvelopeHandler shall be made available and it shall contain the following COMPREHENSION TLVs (the order of the Device Identity object, AID object and File List object given in the system EnvelopeHandler is not specified, the order of the File Update Information objects is relevant):
  - Device Identity with source set to network and destination set to UICC, as defined in ETSI TS 102 223 [7];
  - AID of the ADF, as defined in ETSI TS 102 223 [7], if at least one of the updated files belongs to an ADF. If all successfully updated files belong to the UICC shared file system, the AID TLV object shall not be present;
  - File List, as defined in ETSI TS 102 223 [7]. The number of files shall be set to the number of successful update commands which were applied to files which were registered by the triggered applet. Files which were not registered by the triggered applet shall not occur in the File List. If a file belongs to the ADF indicated in the AID TLV, then its path starts with "3F007FFF". If a file has been updated several times in the command string, then it appears several times in the file list. If SFI referencing is used in the APDU Command, it shall be converted to its File Identifier. In case of an update or increase to a file that is deleted within the same command string the file shall not be included in the File List TLV for this event.
  - File Update Information objects, as defined for EVENT_EXTERNAL_FILE_UPDATE. The system EnvelopeHandler shall contain as many File Update Information objects as there were successful update operations on files listed in the File List object. The order of each File Update Information object shall reflect the order of the files in the File List object.

NOTE 2: The maximum number of updated files that can be managed by the CAT Runtime Environment is implementation dependent.

NOTE 3: In case of many update commands in one RFM command string, the list of COMPREHENSION TLVs may exceed the capacity of the system Envelope Handler. In that case the list of COMPREHENSION TLVs may be truncated.

The registration to this event is effective once the applet has successfully called any of the methods registerFileEvent(…).

The deregistration for a particular file to this event is effective once the Applet has successfully called any of the methods deregisterFileEvent(…). A call to the method *clearEvent*(EVENT_REMOTE_FILE_UPDATE) clears the event EVENT_REMOTE_FILE_UPDATE from the Toolkit Registry of the Applet i.e. the Applet is no longer triggered when a file is updated due to a command in an RFM command string.

EVENT_MEMORY_FAILURE

If the UICC is provided with memory reliability monitoring mechanism (see clause 6.7.2), the CAT Runtime Environment shall trigger all the Toolkit applets registered to this event when the card operating system has detected an irrecoverable memory failure in any location of the persistent memory.

An irrecoverable memory failure is said to occur when a loss of data is detected or when it is no longer possible to write data to memory. The event is reported once per applet, during the lifecycle of the card.

The memory reliability monitoring mechanism and its implementation specific limitations are described in clause 6.7.2.

EVENT_TERMINAL_APPLICATIONS

Upon reception of an ENVELOPE (TERMINAL APPLICATIONS) APDU command as defined in ETSI TS 102 221 [6] with the ENVELOPE (TERMINAL APPLICATIONS) as defined in ETSI TS 102 223 [7] the CAT Runtime Environment shall trigger the Toolkit applets registered to the corresponding event.

EVENT_POLL_INTERVAL_NEGOTIATION

Upon reception of an ENVELOPE (POLL INTERVAL NEGOTIATION) APDU command as defined in ETSI TS 102 221 [6] the CAT Runtime Environment shall trigger all Toolkit applets registered to this event.

Toolkit applets can request a poll interval shorter than the poll interval suggested by the Terminal by invoking the method *uicc.toolkit.ToolkitRegistry.requestPollIntervall(short duration)*. Requests for longer values shall be ignored by the CAT Runtime Environment.

6.3 Registration

A Toolkit applet shall register to the JCRE as specified in "Runtime Environment Specification, Java Card™ Platform, 3.0.1 Classic Edition" [3].

A Toolkit applet shall register to the CAT Runtime Environment, 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 CAT Runtime Environment 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 CAT Runtime Environment so that the Toolkit applets can:

- initialize a proactive command with the *init()* method;
- append several Simple TLV as defined in ETSI TS 102 223 [7] to the proactive command with the *appendTLV()* methods;
- request the CAT Runtime Environment 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 CAT Runtime Environment 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 CAT Runtime Environment 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 CAT Runtime Environment 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 CAT Runtime Environment shall throw an exception.

The CAT Runtime Environment 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 CAT Runtime Environment shall throw an exception.

The CAT Runtime Environment shall prevent a Toolkit applet from sending a DECLARE SERVICE (add, delete) proactive command using a service identifier, which is not allocated to it. If an applet attempts to send such a command, the CAT Runtime Environment shall throw an exception.

The CAT Runtime Environment 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 CAT Runtime Environment shall throw an exception.

The CAT Runtime Environment shall prevent a Toolkit applet from sending an OPEN CHANNEL proactive command if it exceeds the maximum number of channels allocated to this applet. If an applet attempts to send such a command the CAT Runtime Environment 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 CAT Runtime Environment.

The CAT Runtime Environment cannot guarantee if the SET UP IDLE MODE TEXT proactive command is used by a Toolkit applet, that 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 CAT Runtime Environment so that the Toolkit applets can send a response to some specific events.
(e.g. EVENT_CALL_CONTROL_BY_NAA). The COMPREHENSION-TLV list contained in the *EnvelopeResponseHandler* shall be sent as the response data of the ENVELOPE command. The Boolean parameter passed to the *post()* or *postAsBERTLV()* method shall be mapped by the CAT Runtime Environment to the correct status word, if the value is true it corresponds to a successful ending of the command status word "9000", if the value is false it corresponds to a warning status word "6200". An extension of the CAT Runtime Environment for a specific NAA can overwrite this mapping.

In case of EVENT_CALL_CONTROL_BY_NAA, the Boolean *value* parameter passed to the *post()* or *postAsBERTLV()* method is meaningless and shall be ignored by the CAT Runtime Environment.

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 CAT Runtime Environment 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 "Runtime Environment Specification, Java Card™ Platform, 3.0.1 Classic Edition" [3].

A system handler is available if the exception *ToolkitException*. HANDLER_NOT_AVAILABLE is not thrown when the corresponding *getTheHandler()* method is called or a method of its interface is called.

A system handler shall not be available if the corresponding *getTheHandler()* method is not called, directly or indirectly, from the applet's *processToolkit*() method. If necessary and only when explicitly stated in another specification, the *ProactiveHandler* and the *ProactiveResponseHandler* may in addition be available if the corresponding *getTheHandler()* method was called from within a different method than *processToolkit()*.

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. These rules apply also, if the *ProactiveHandler* or the *ProactiveResponseHandler* are available in a method different from *processToolkit()*. Under this condition, and for all following rules concerning the *ProactiveHandler* and the *ProactiveResponseHandler*, the method name "processToolkit()" has to be replaced by the method name in which the method *getTheHandler()* was called. The following rules concerning the *ProactiveHandler* and the *ProactiveResponseHandler* may in addition be modified by another specification.

ProactiveHandler:

- The *ProactiveHandler* shall not be available if the Terminal Profile command has not yet been processed by the CAT Runtime Environment.
- 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 CAT Runtime Environment.

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 remain 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.

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 (see note 2)	Envelope Handler	Envelope Response Handler	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 1)
_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 1)
_CHANNEL_STATUS	Y	Y	N	1/7 (per channel) (see note 1)
_ACCESS_TECHNOLOGY_CHANGE	Y	Y	N	n/n
_ACCESS_TECHNOLOGY_CHANGE_ MULTIPLE	Y	Y	N	n/n
_DISPLAY_PARAMETER_CHANGED	Y	Y	N	n/n
_NETWORK_SEARCH_MODE_CHANGE	Y	Y	N	n/n
_BROWSING_STATUS	Y	Y	N	n/n
_FRAMES_INFORMATION_CHANGED	Y	Y	N	n/n
_HCI_CONNECTIVITY	Y	Y	N	n/n
_LOCAL_CONNECTION	Y	Y	N	1/8 (per service identifier) (see note 2)
_CONTACTLESS_STATE_REQUEST	Y	Y	N	n/n
_POLL_INTERVAL_NEGOTIATION	N	Y	N	n/n
_TERMINAL_APPLICATIONS	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
_PROACTIVE_HANDLER_AVAILABLE	N	N	N	n/n
_FIRST_COMMAND_AFTER_ATR	N	N	N	n/n
_EXTERNAL_FILE_UPDATE	N	Y	N	n/n
_APPLICATION_DESELECT	N	Y	N	n/n
_REMOTE_FILE_UPDATE	N	Y	N	n/n
_MEMORY_FAILURE	N	N	N	n/n

NOTE 1: One Toolkit applet can register to several timers/channels/services identifier, but a timer/channel/services identifier can only be allocated to one Toolkit applet.

NOTE 2: It is recommended to use ISOException with reason code 0x9300 only for events where reply busy is allowed.

6.7 CAT Runtime Environment behaviour

6.7.0 Basic rules

The following rules define the CAT Runtime Environment behaviour for:

- ToolkitInterface object retrieval:
  - The CAT Runtime Environment shall invoke the *getShareableInterfaceObject()* method of the Toolkit applet to retrieve the reference of its *ToolkitInterface* object, before triggering it the first time in its life cycle.
  - The AID parameter of the *getShareableInterfaceObject()* method shall be set to null.
  - The byte parameter of the *getShareableInterfaceObject()* method shall be set to one (i.e. "01").
- Triggering of a Toolkit applet (invocation of the *processToolkit()* method of the *ToolkitInterface*object):
  - The CAT Runtime Environment 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 "Runtime Environment Specification, Java Card™ Platform, 3.0.1 Classic Edition" [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.
  - The context as defined in Java Card™ shall be set to the context of the Toolkit applet. The previous context (context of the caller) shall be the context of the CAT Runtime Environment.
- 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

6.7.1.0 Overall behaviour

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

The CAT Runtime Environment 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 CAT Runtime Environment 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 CAT Runtime Environment shall send a SET UP MENU system proactive command whenever a menu entry is modified, added or removed or the EF_{SUME} file under the $DF_{TELECOM}$ file is updated as defined in ETSI TS 102 222 [8].

If help is available for at least one Menu Entry inserted in the SET UP MENU system proactive command the CAT Runtime Environment shall indicate to the terminal that help information is available. Otherwise the CAT Runtime Environment shall not indicate to the terminal that help information is available.

The CAT Runtime Environment shall use the data of the EF_{SUME} file under the DF_Telecom when issuing the SET UP MENU proactive command.

If a text attribute different from the default format is provided for at least one Menu Entry, the SET UP MENU system proactive command shall contain the item text attribute list Comprehension TLV. The default format as defined in ETSI TS 123 040 [12] is "00 00 03 90".

A Menu Entries' list is managed by the CAT Runtime Environment. The Menu Entries' list is a simple link list which is modified either when *initMenuEntry()* is successfully called or when an applet is successfully deleted. The Menu Entries' list is managed regardless of the menu entry state (enable/disable) as well as regardless of the Toolkit applet(s) life cycle state(e.g. Selectable/Locked, etc.).

Each element of the list corresponds to an Item used by the CAT Runtime Environment to build and send the SET UP MENU system proactive command to the terminal. The CAT Runtime Environment shall provide the items to the terminal in the same order than in the Menu Entries' list (from the first element to the last element).

The positions of the Toolkit applet menu entries in the Menu Entries' 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 (ETSI TS 102 226 [10]) and provided to the CAT Runtime Environment.
  - The range (128,255) is managed by the CAT Runtime Environment. When the requested item identifier is "00" the CAT Runtime Environment shall assign the first free value in the range (128,255).
- Item position: The Item position of a Menu Entry indicates the position where the Menu Entry shall be inserted in the Menu Entries' list.
  - If the new Menu Entry has to be inserted at an already occupied position, the entries from the requested position to the last element of the Menu Entries' list are shifted to the next positions.
  - If the position indicated is greater than the number of elements in the Menu Entries' list, then the Menu Entry takes the last position in the Menu Entries' list.
  - If the position indicated is equal to "00", then the Menu Entry takes the last position in the Menu Entries' list.

6.7.1.2 SET UP EVENT LIST

At the beginning of a CAT session, the CAT Runtime Environment shall send a SET UP EVENT LIST system proactive command, if at least one of the EVENT_EVENT_DOWNLOAD_* events is registered by a selectable Toolkit applet.

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

6.7.1.4 NEGOTIATION OF POLL INTERVAL

If at least one Toolkit applet has registered the EVENT POLL INTERVAL NEGOTIATION the Terminal can send an ENVELOPE(POLL INTERVAL NEGOTIATION) at any time.

The CAT Runtime Environment shall send a response to the EVENT POLL INTERVAL NEGOTIATION as defined in ETSI TS 102 223 [7] according to the following rules:

If none of the triggered Toolkit applets has requested a shorter poll interval duration than the one supplied by the Terminal, and if none of the Toolkit applets that is not triggered on this event has previously requested a shorter poll interval duration than the one supplied by the Terminal, the CAT Runtime Environment shall respond with the "Poll interval result" set to "Accepted" as defined in ETSI TS 102 223 [7] and the "Duration" set to the new poll interval duration.

NOTE: Terminals compliant with TS 102 223 Rel-12, Rel-13 or Rel-14 do not expect the "Duration" object in this situation and may show unpredictable behaviour if the object is present.

If one or more triggered applets requested a shorter poll interval duration than the poll interval duration value supplied by the Terminal, or if at least one of the Toolkit applets that is not triggered on this event has previously requested a shorter poll interval duration than the one supplied by the Terminal, the CAT Runtime Environment shall respond with the "Poll interval result" set to "Modified" and with the lowest poll interval value duration requested by the Toolkit applets set as "Duration" as defined in ETSI TS 102 223 [7].

The "Duration" sent in the response to the ENVELOPE(POLL INTERVAL NEGOTIATION) shall be persistent and is the new value of the poll interval duration used by the CAT Runtime Environment according to the rules defined in clause 6.7.1.3, and as the response to an invocation of the *ToolkitRegistry.getPollInterval()* method.

6.7.1.5 ACTIVATE

This clause applies if the UICC supports ETSI TS 102 613 [17] and ETSI TS 102 705 [18].

The CAT Runtime Environment shall send this system proactive command automatically only if all the following conditions are fulfilled:

- the terminal supports the SWP [17] interface and indicates support of the ACTIVATE proactive command in the Terminal Profile;
- the UICC supports the `AUTO_ACTIVATE_SERVICE_ID` service as defined in [18]; and
- the SWP [17] interface is in DEACTIVATED state; and
- the UICC needs to communicate using this interface.

The CAT Runtime Environment shall not prevent applets from sending this proactive command.

6.7.2 UICC memory reliability monitoring

The support of the event EVENT_MEMORY_FAILURE is optional.

If a Toolkit applet tries to register to this event, and the event is not supported, the registration shall fail, and a ToolkitException is generated, with EVENT_NOT_SUPPORTED reason code.

Typical persistent memory technologies suffer of limited write cycles. Silicon manufacturers specify a nominal minimum guaranteed number of writecycles at specific conditions. When this number is exceeded for a memory cell or operating conditions are extreme, persistent memory reliability degrades and memory operations may fail.

In UICCs, particularly for those destined to M2M applications, the card operating system may be provided with an optional mechanism for monitoring the status of persistent memory or part of it.

Using said mechanism, the CAT Runtime Environment can inform the applets sending them the event EVENT_MEMORY_FAILURE. The applets triggered by this event are responsible to perform recovery actions, e.g. by remotely signalling the card Issuer that a replacement is advised. The memory reliability monitoring mechanism could be based on either hardware facilities, on software solutions or both. The techniques implemented to detect defective memory cells are card manufacturer specific and out of scope of the present document.

It has to be understood that there are certain limitations of the aforementioned mechanism:

- If the UICC is provided with further mechanism to recover from write errors, e.g. by multiple writing attempts, by reallocating data structures to different addresses etc., errors that can be transparently recovered by the UICC shall not result in the triggering of the aforementioned event. Recovering techniques are manufacturer dependant and are not mandated by the present document.
- The techniques to deal with memory failure may depend on the support of the semiconductor device manufacturer and are UICC manufacturer specific and out of scope of the present document.
- There is no guarantee that the aforementioned event will be sent to an application. A sudden memory failure could affect a part of memory vital to the card operating system, the CAT Runtime Environment and/or Java Card™ Runtime Environment any time.

7 Toolkit applet

7.1 Applet loading

The UICC API card shall be compliant to the "Virtual Machine Specification Java Card™ Platform, 3.0.1 Classic Edition" [4] and to annex B to guarantee interoperability at byte code Level.

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

7.2 Data and function sharing

The sharing mechanism defined in "Application Programming Interface, Java Card™ Platform, 3.0.1 Classic Edition" [2] and "Runtime Environment Specification, Java Card™ Platform, 3.0.1 Classic Edition" [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 "Runtime Environment Specification, Java Card™ Platform, 3.0.1 Classic Edition" [3] shall be used to delete the content from the UICC. The object deletion mechanism defined optional in "Application Programming Interface, Java Card™ Platform, 3.0.1 Classic Edition" [2] is mandatory.

If requested by an applet, the object deletion shall start prior to the processing of the next APDU if no applet is running or suspended. This implies that it cannot be guaranteed that the object deletion has been performed prior to the next invocation of the *applet.process() method* or *ToolkitInterface.processToolkit()* method.

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 s could be required. During this period the UICC should respect the work waiting time procedure, defined in ETSI TS 102 221 [6].

8 UICC and ADF File System Administration API

8.0 Overview

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

8.1 AdminFileView objects

The interface *AdminFileView* extends the interface *FileView*, i.e. objects implementing the interface *AdminFileView* inherit *FileView* functionality.

An *AdminFileView* object can be retrieved by invoking one of the *getAdminFileView()* methods defined in the *AdminFileViewBuilder* class.

If BER TLV files functions are supported by an implementation, the *getAdminFileView()* and *getTheUICCAdminFileView()* methods defined in the *AdminFileViewBuilder* class shall return the reference of an object implementing the *AdminBERTLVFileView* interface.

Each *AdminFileView* shall be given the access control privileges associated with the UICC or the corresponding ADF for the applet. The access control privileges are defined by the UICC Administrative access application specific parameters specified in ETSI TS 102 226 [10]. UICC access application specific parameters shall not apply to objects retrieved from the *uicc.access.fileadministration.AdminFileViewBuilder* class. The access control privileges are checked against the access rules defined in ETSI TS 102 221 [6] each time a method of the *AdminFileView* object is invoked.

8.2 AdminFileView operations

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

- CREATE FILE as defined in ETSI TS 102 222 [8]. Creation of an ADF at the API level is FFS.
- DELETE FILE as defined in ETSI TS 102 222 [8].
- RESIZE as defined in ETSI TS 102 222 [8].

9 UICC Java Card™ Services

9.0 Introduction

UICC Java Card™ Services are implemented as GlobalPlatform Global Services Applications according to the GlobalPlatform Card Specification [15]. A unique service name identifies each service. Applets request a reference to a UICC Java Card™ Service through the following method defined in the GlobalPlatform API [16]:

```
org.globalplatform.GPSystem.getService(javacard.framework.AID serverAID, short sServiceName)
```

The service names constant values are defined in the *uicc.system.servicesConstants* interface . The support for any of the UICC Java Card™ Services defined in the present document is optional. In case a specific service is not supported, the *getService()* method invoked with the corresponding service name shall return *null*.

9.1 High update arrays

The *uicc.services.highupdatearray.HighUpdateArrayBuilder* shareable interface is an optional UICC Java Card™ Service that provides the creation of Java Card™ arrays, called high update arrays, which support a specified number of update operations, which is expected to be higher than required for general data. Each update action on the array, regardless of the position and number of the updated element(s), is counted as one update operation. The service name to be used to obtain a reference to the *HighUpdateArrayBuilder* object is *SERVICE_ID_HIGH_UPDATE_ARRAY_BUILDER*.

The unavailability of this Service does not exclude that the platform may perform the management of frequently updated data transparently. In that case, the application may use the standard array feature.

If the high update arrays service is available, the UICC shall be classified with the JX property representing its update performance with reference to high update arrays, in a similar way as described in ETSI TS 102 671 [14]. The JX property value indicates the UICC's expected minimum number of update operations supported for a high update array. The following JX property values are defined:

- JA: UICCs indicating JA as their minimum number of update operations property shall be able to update an high update array 100 000 times without failure; loss of information due to time factors is excluded from this property.
- JB: UICCs indicating JB as their minimum number of update operations property shall be able to update an high update array 500 000 times without failure; loss of information due to time factors is excluded from this property.
- JC: UICCs indicating JC as their minimum number of update operations property shall be able to update an high update array 1 000 000 times without failure; loss of information due to time factors is excluded from this property.

Applets can query the JX property value of the UICC by using the method *HighUpdateArrayBuilder.makeHighUpdateObjectArray()*.

10 UICC Java Card Runtime Environment

10.1 Overview

The UICC Java Card Runtime Environment is an extension of the "Runtime Environment Specification, Java Card™ Platform" described in [3]. The UICC Runtime Environment offers services not related to the CAT Runtime Environment by means of a dedicated API that extends "Application Programming Interface, Java Card™ Platform" [2].

10.2 UICC suspension

10.2.1 UICC Suspension purpose

The UICC suspension mechanism allows the terminal to suspend the UICC when access is not required a long period of time. Upon suspension the UICC shall store its internal state and volatile data in order to restore them upon a successful resume operation.

The interface *uicc.suspendresume.SuspendMechanism* allows applets to:

- be informed of a suspend request of the terminal;
- reject the request to suspend the UICC;
- accept the suspend request, with indication of the maximum suspension time for the applet;
- apply its own logic prior its suspension;

- be informed of and to apply its own logic after a successful resume.

Applets that do not implement this interface do not take part in this mechanism and are suspended by the UICC Runtime Environment without notification.

Exceptions raised by applets during the execution of methods defined in the interface *uicc.suspendresume.SuspendMechanism* shall not be propagated to the terminal.

Upon the invocation of all methods of *uicc.suspendresume.SuspendMechanism* except explicitly stated, the UICC Runtime Environment shall invoke the applet as the "*currently selected applet instance*" according to [3].

10.2.2 Suspension mechanism

10.2.2.1 Suspension mechanism overview

On reception of a SUSPEND UICC command for suspend operation, as described in ETSI TS 102 221 [6], the UICC Runtime Environment shall execute the following steps:

- evaluate the request, to determine if the suspension is possible and to determine the maximum time, according to clause 10.2.2.2;
- inform the applets about the suspension and allow them to apply any required action according to clause 10.2.2.3;
- process the suspension.

10.2.2.2 Suspension Request Operation

To properly evaluate the suspension request, the UICC Runtime Environment shall call *suspendRequest()* for all applets that implement the *uicc.suspendresume.SuspendMechanism* interface with the minimum and maximum time value received in the SUSPEND UICC command. Each applet shall evaluate if the request can be accepted based on the input parameters provided by the method.

If an applet rejects the suspension, either by raising an exception, by answering with an interval time lower than the proposed interval time or by answering with an incorrect time interval unit, the UICC Runtime Environment shall not continue to call the method *suspendRequest()* of the remaining applets and shall reject the suspension mechanism request returning to the command the status word '9864'.

If no applet rejects the suspension, the UICC Runtime Environment determines the maximum duration of the suspension taking the lowest time value returned by all applets. If all applets provide a value higher than the allowed maximum value, the UICC Runtime Environment shall select the allowed maximum value.

10.2.2.3 Suspension Operation

The UICC Runtime Environment shall process the suspension mechanism calling the method *suspendOperation()* for all applets that implement the *uicc.suspendresume.SuspendMechanism* interface with the maximum suspension time computed from Suspension Request Operation.

If an applet raises an exception, the suspension operation shall not be stopped. The UICC Runtime Environment shall not propagate the exception.

Afterwards, the UICC Runtime Environment shall store the UICC context to be suspended composed of its internal state and volatile data. The UICC Runtime Environment shall not deselect applets selected on any logical channels.

The UICC Runtime Environment shall answer to the APDU command with the maximum suspension time allowed and with the generated 8 byte token.

10.2.3 Resume mechanism

10.2.3.1 Resume mechanism overview

Upon reception of a SUSPEND UICC command for resume operation, as described in ETSI TS 102 221 [6], the UICC evaluate the command. If the command is accepted the UICC Runtime Environment shall execute the following steps:

- restore the UICC suspended context with all its internal state and all applets states;
- notify applets of the restoration of the their context according to clause 10.2.2.2;
- delete the UICC suspended context.

If the command is rejected, the UICC Runtime Environment shall delete the UICC suspended context and the CAT Runtime Environment shall trigger Toolkit applets on queued events according to clause 6.1.0.

10.2.3.2 Resume Indication

Afterward, the UICC Runtime Environment shall call the method *resumeIndication()* of all applets that implement the *uicc.suspendresume.SuspendMechanism*.

If an applet raises an exception, the UICC Runtime Environment shall not propagate the exception and shall not cancel the resume operation, but it shall continue to call the method *resumeIndication()* of remaining applets.

10.2.4 Handler management

For all methods of the *uicc.suspendresume.SuspendMechanism* interface, the CAT Runtime Environment shall not allow access to system handlers. ProactiveHandler, ProactiveResponseHandler, EnvelopeHandler and EnvelopeResponseHandler shall not be available.

Annex A (normative): Java Card™ UICC API

The source files for the Java Card™ UICC API (102241_Annex_A_Java.zip and 102241_Annex_A_HTML.zip) are contained in ts_102241v140100p0.zip, which accompanies the present document.

Annex B (normative): Java Card™ UICC API identifiers

The export files for the uicc.* package (102241_Annex_B_Export_Files.zip) are contained in ts_102241v140100p0.zip, which accompanies the present document.

NOTE: See the "Virtual Machine Specification Java Card™ Platform, 3.0.1 Classic Edition" [4].

Annex C (normative): UICC API package version management

Table C.1 describes the relationship between each ETSI TS 102 241 [13] specification version and its UICC API packages AID and Major, Minor versions defined in the export files.

Table C.1

ETSI TS 102 241 [13]	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	1.2	A0 00 00 00 09 00 05 FF FF FF FF 89 12 00 00 00	1.11

Table C.2

ETSI TS 102 241 [13]	uicc.system package	
	AID	Major, Minor
	A0 00 00 00 09 00 05 FF FF FF FF 89 13 00 00 00	1.1

Table C.3

ETSI TS 102 241 [13]	uicc.access.fileadministration package	
	AID	Major, Minor
	A0 00 00 00 09 00 05 FF FF FF FF 89 11 01 00 00	1.0

Table C.4

ETSI TS 102 241 [13]	uicc.access.bertlvfile package	
	AID	Major, Minor
	A0 00 00 00 09 00 05 FF FF FF FF 89 11 02 00 00	1.0

Table C.5

ETSI TS 102 241 [13]	uicc.services.highupdatearray package	
	AID	Major, Minor
11.1.0	A0 00 00 00 09 00 05 FF FF FF FF 89 18 01 00 00	1.0

Table C.6

ETSI TS 102 241 [13]	uicc.suspendresume package	
	AID	Major, Minor
14.0	A0 00 00 00 09 00 05 FF FF FF FF 89 14 00 00 00	1.0

The package AID coding is defined in ETSI 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.

Annex D (informative): Menu order example

D.0 Preamble

The following examples are in consecutive order.

D.1 State after initialization

Position in ToolkitRegistry Menu Entries' list	Name	SET UP MENU proactive command

D.2 Some application installation later

Position in ToolkitRegistry Menu Entries' list	Name	SET UP MENU proactive command
1	Legacy1	Legacy1
2	Legacy2	Legacy2
3	Legacy3	Legacy3
4	Legacy4	Legacy4

D.3 Installation of application A with position of menu entry set to 3

Position in ToolkitRegistry Menu Entries' list	Name	SET UP MENU proactive command
1	Legacy1	Legacy1
2	Legacy2	Legacy2
3	A	A
4	Legacy3	Legacy3
5	Legacy4	Legacy4
NOTE: The indicated position 3 pushes the entries "Legacy3" and "Legacy4" one position down.		

D.4 Installation of application B with position of menu entry set to 3

Position in ToolkitRegistry Menu Entries' list	Name	SET UP MENU proactive command
1	Legacy1	Legacy1
2	Legacy2	Legacy2
3	B	B
4	A	A
5	Legacy3	Legacy3
6	Legacy4	Legacy4
NOTE: The indicated position 3 pushes also the previously installed Application A from position 3 one position down to the new position 4.		

D.5 Installation of application C with position of menu entry set to 2 and 3

D.5.1 Insert at position 2

Position in ToolkitRegistry Menu Entries' list	Name	SET UP MENU proactive command
1	Legacy1	Legacy1
2	C1	C1
3	Legacy2	Legacy2
4	B	B
5	A	A
6	Legacy3	Legacy3
7	Legacy4	Legacy4

D.5.2 Insert at position 3

Position in ToolkitRegistry Menu Entries' list	Name	SET UP MENU proactive command
1	Legacy1	Legacy1
2	C1	C1
3	C2	C2
4	Legacy2	Legacy2
5	B	B
6	A	A
7	Legacy3	Legacy3
8	Legacy4	Legacy4

D.6 Installation of application D with position of menu entry set to "00"

Position in ToolkitRegistry Menu Entries' list	Name	SET UP MENU proactive command
1	Legacy1	Legacy1
2	C1	C1
3	C2	C2
4	Legacy2	Legacy2
5	B	B
6	A	A
7	Legacy3	Legacy3
8	Legacy4	Legacy4
9	D	D

D.7 Installation of application E with position of menu entry set to 20

Position in ToolkitRegistry Menu Entries' list	Name	SET UP MENU proactive command
1	Legacy1	Legacy1
2	C1	C1
3	C2	C2
4	Legacy2	Legacy2
5	B	B
6	A	A
7	Legacy3	Legacy3
8	Legacy4	Legacy4
9	D	D
10	E	E

D.8 Disabling/Locking of application legacy1 and application A with menu entries at position 1 respectively 6

Position in ToolkitRegistry Menu Entries' list	Name	SET UP MENU proactive command
1	Legacy1	C1
2	C1	C2
3	C2	Legacy2
4	Legacy2	B
5	B	Legacy3
6	A	Legacy4
7	Legacy3	D
8	Legacy4	E
9	D	
10	E	

D.9 Re-enabling/Unlocking of application legacy1 and application A with menu entries at position 1 respectively 6

Position in ToolkitRegistry Menu Entries' list	Name	SET UP MENU proactive command
1	Legacy1	Legacy1
2	C1	C1
3	C2	C2
4	Legacy2	Legacy2
5	B	B
6	A	A
7	Legacy3	Legacy3
8	Legacy4	Legacy4
9	D	D
10	E	E

D.10 Deletion of application A with menu entry at position 6

Position in ToolkitRegistry Menu Entries' list	Name	SET UP MENU proactive command
1	Legacy1	Legacy1
2	C1	C1
3	C2	C2
4	Legacy2	Legacy2
5	B	B
6	Legacy3	Legacy3
7	Legacy4	Legacy4
8	D	D
9	E	E
NOTE: Menu entries below menu position 6 are moved up one position.		

Annex E (informative): Change history

This annex lists all Changes Requests (CR) applied to the present document.

Meeting	Plenary Tdoc	VERS	CR	REV	CAT	SUBJECT	Resulting Version
SCP-14	SCP-030212	6.0.0	003		B	Menu Entries position management	6.1.0
SCP-15	SCP-030483	6.1.0	008	1	B	API to react on the end of a Proactive Session	6.2.0
	SCP-030484		009	1	C	API correction to be Transport Protocol independent	
	SCP-030454		015		C	Upgrade the reference from Java Card™ 2.2 to version 2.2.1	
	SCP-030454		004		C	New method appendTLV() with two byte arrays as input parameter.	
	SCP-030454		006		B	Add new methods initMoreTime() in class ProactiveHandler	
	SCP-030454		007		B	Introduction of Global Byte Array	
	SCP-030454		010		B	Specification of the first command after ATR event	
	SCP-030454		011		D	ProactiveResponseHandlerSystem.getTheHandler() method set to public	
	SCP-030454		012		D	Incorrect wording in UICCException	
	SCP-030454		014	1	B	Introduction of BER and COMPREHENSION TLV Handlers	
SCP-16	SCP-040047	6.2.0	005	1	B	Addition of select(SFI) method	6.3.0
	SCP-040068		016	1	C	getTheFileView throw ArrayIndexOutOfBoundException when an AID is passed as byte array with invalid offset and length parameters	
	SCP-040069		017	1	C	Issuing system proactive command SET UP MENU in case EF_{SUME} is updated	
	SCP-040047		018		F	Update of UICC Java Card™ Architecture diagram	
	SCP-040047		019		C	Clarification of CAT Runtime Environment behaviour	
	SCP-040071		020	1	C	Specification of Java Card™ object deletion for UICC Java Card™ and Toolkit applet	
	SCP-040067		021	1	C	Modification of LOCAL SERVICE identifiers management	
	SCP-040047		023		D	Renaming of the attached files	
	SCP-040047		025		D	Remove all references to 51.011	
SCP-17	SCP-040214	6.3.0	027		F	Reordering of UICCException reason codes	6.4.0
	SCP-040214		028		F	Suppression of FILE_INVALIDATED reason code	
	SCP-040214		029		C	Splitting of the proprietary range of events	
	SCP-040214		013	2	C	Allow passing of specified status words through the toolkit framework	
	SCP-040214		032		C	Specify the system handlers availability outside of processToolkit() invocation	
	SCP-040214		033		D	Update clauses where HANDLER_NOT_AVAILABLE reason is used.	
	SCP-040214		037		D	Editorial cleaning	
	SCP-040271		038		D	Addition of text formatting for menu items	
	SCP-040277		030		F	Clarification of EVENT_UNRECOGNIZED_ENVELOPE definition	
	SCP-040278		031	1	F	Clarify behaviour upon an unsuccessful TLV search	
	SCP-040279		034	1	B	Introduction of Browsing status event and Network search mode change event	
	SCP-040280		035	1	F	Clarification of the Access Controls for the File Access API	
	SCP-040281		036	1	B	Introduction of an API to create, delete and resize files	
	SCP-040289		040		B	Introduction of File Event	
SCP-18	SCP-040311	6.4.0	041		F	Correction to constructor of HandlerBuilder class of uicc.system package	6.5.0
			042		C	Remove getValue(short idx) method in TerminalProfile class of uicc.toolkit package	
	SCP-040365		043		F	Addition of exceptions in ViewHandler buildTLVHandler() methods definition	
	SCP-040311		044		F	Clarification of EVENT_PROACTIVE_HANDLER_AVAILABLE registration	
			045		D	Clarifications in documentation of method uicc.access.FileView.searchRecord()	

Meeting	Plenary Tdoc	VERS	CR	REV	CAT	SUBJECT	Resulting Version
			046		F	Clarification about capacity parameter of buildTLVHandler() methods	
			048		F	Correction of erroneous constant definitions in uicc.access.UICCConstants.java	
SCP-19	SCP-040432	6.5.0	049		F	Clarification for non-specific references	6.6.0
			050		F	Terminal Profile update to the latest changes in ETSI TS 102 223	
			052		F	Definition of TAR_NOT_DEFINED for ToolkitException	
			053		F	Clarification for Exception in case capacity is negative	
			054		F	Clarification for the *EVENT_EXTERNAL_FILE_UPDATE*	
			055		F	Terminal Profile update for text attribute features	
SCP-19	SCP-040432	6.6.0	051		D	Clarification in description of AdminFileView	7.0.0
SCP-20	SCP-050019	7.0.0	057		A	Corrections in documentation of Java methods for file event registration	7.1.0
			059		A	Access rights clarification for FileView and AdminFileView	
			061		A	Correction of SET UP EVENT LIST system command behaviour	
			063		A	Clarification of file event deregistration	
SCP-22	SCP-050244	7.1.0	065		A	Clarification of envelope response handling in case of EVENT_CALL_CONTROL_BY_NAA	7.2.0
			067		A	Addition of missing OUT_OF_TLV_BOUNDARIES ToolkitException in getChannelIdentifier() method definition of ProactiveResponseHandler interface	
			069		F	Delete the reference to ISO/IEC 7816-3	
	SCP-050231		077		A	Addition of missing AdminException.INCORRECT_PARAMETERS exception in resizeFile() method definition	
			079		A	Correction of description for SET UP MENU	
SCP-23	SCP-050484	7.2.0	071	1	A	Clarifications and corrections in FileView interface of uicc.access package	7.3.0
			080		D	Corrections in the description of the method HandlerBuilder.buildTLVHandler()	
			087		B	Reservation of events values "121" and "122" for 3GPP	
			089		A	Clarifications and corrections in FileView interface of uicc.access package	
			090		D	Corrections createFile and resizeFile method description	
	SCP-050493		086		A	Clarify handler availability for EVENT_APPLICATION_DESELECT	
	SCP-050500		087		B	Addition of UICCException reason code CONDITIONS_OF_USE_NOT_SATISFIED	
	SCP-050501		084		B	Define the constant for the proactive command send short message	
SCP-25	SCP-060154	7.3.0	092	1	A	UICC API increase	7.4.0
SCP-26	SCP-060286	7.4.0	095		D	Correct a comment about TAG_FCP_LCS_INTEGER value	7.5.0
			096		B	Event External File Update : support for BER-TLV files	
	SCP-060283		094	2	B	Introduction of new exceptions to reflect changes due to the introduction of the termination state for files in ETSI TS 102 221 and ETSI TS 102 222	
SCP-27	SCP-060445	7.5.0	101		A	Correction of the release for references	7.6.0
	SCP-060476		097	1	F	Reserve a short identifier for a 3GPP event defined in ETSI TS 102 223	
SCP-29	SCP-070023	7.6.0	106		A	Correction of method AdminFileView.resizeFile() for BER-TLV Files	7.7.0
			108		F	Correction of incorrect constant value in UICCConstants.java	
			109		B	Reference to Java Card™ 2.2.2 specification	
SCP-30 bis	SCP-070191	7.7.0	099	2	B	Support for RETRIEVE DATA and SET DATA functions for BER-TLV files	7.8.0 withdrawn
			104	2	B	Addition of a method for concurrent card application toolkit sessions	
						Missing values supplied in COMMAND_NOT_ALLOWED, class uicc.access.UICCException (in attachment).	

Meeting	Plenary Tdoc	VERS	CR	REV	CAT	SUBJECT	Resulting Version
ETSI TS 102 241 v7.8.0 was withdrawn (decision made at SCP #35)							
SCP-30 bis	SCP-070191	7.7.0	099	2	B	Support for RETRIEVE DATA and SET DATA functions for BER-TLV files	7.9.0
						Missing values supplied in COMMAND_NOT_ALLOWED, class uicc.access.UICCException (in attachment).	
SCP-33	SCP-070419	7.7.0	111		F	Modification of CAT Runtime Environment behaviour in case of CLOSE CHANNEL command for UICC Server Mode in mode "TCP in LISTEN state"	
			112		D	Editorial Correction in Method uicc.system.HandlerBuilder.buildTLVHandler()	
SCP-35	SCP-070191	7.7.0	104	2	B	Addition of a method for concurrent card application toolkit sessions (Only changes to clauses 6.1 and 6.2 are implemented as the rest of the changes is superseded by the technical content in CR 113)	7.9.0
SCP-35	SCP-080035	7.9.0	113	1	F	Implement the isPrioritizedProactiveHandlerAvailableEventSet method to the "ToolkitRegistrySystem" class	8.0.0
SCP-40	SCP-090067	8.0.0	115	1	F	Addition of missing CAT events in table 1 (wrong value allocated for EVENT_EVENT_DOWNLOAD_NETWORK_REJECTION - new value allocated with the rapporteur)	8.1.0
SCP-41	SCP-090125	8.0.0	116		B	Availability of the ProactiveHandler and the ProactiveResponseHandler	8.1.0
SCP-44	SCP(10)0024	8.0.0	117		F	Reservation of Event value for 3GPP CT6	8.1.0
SCP-45	SCP(10)0182	8.1.0	118		F	Change reference from 'Java Card 2.2.2' to 'Java Card 3.0.1 Classic Edition'	9.0.0
SCP-47	SCP(11)0044	9.0.0	123		F	Addition of missing contactless state request event (CR number renumbered from 118 to 123)	9.1.0
SCP-48	SCP(11)0096	9.0.0	120		F	CR 102 241 R9 #120: Correction to ToolkitConstants.java	9.1.0
SCP-48	SCP(11)0097	9.0.0	121		F	CR 102 241 R9 #121: Correction to TerminalProfile Interface	9.1.0
SCP-48	SCP(11)0098	9.0.0	122		F	CR 102 241 R9 #122: Correction of packages version	9.1.0
SCP-51	SCP(11)0227r1	9.1.0	125	1	A	Correction of package versions	9.2.0
SCP-51	SCP(11)0228r1	9.1.0	126	1	F	Adding constant values for contactless operation and other features	9.2.0
SCP-51	SCP(11)0265r1	9.2.0	127		F	Reservation of IMS events for 31.130	10.0.0
SCP-52	SCP(11)0281r1	9.2.0	129		A	Corrections related to changes of event Access Technology Change in ETSI TS 102 223 (mirror of SCPTEC(11)0137)	10.0.0
SCP-53	SCP(11)0377	10.0.0	130		B	M2M Events for monitoring of data reliability	11.0.0
SCP-57	SCP(12)000261	11.0.0	131		B	API services for high activity arrays	11.1.0
SCP-61	SCP(13)000237	11.0.0	132		F	Adding constants for the TerminalProfile interface	11.1.0
SCP-68	SCP(15)000122	11.0.0	137		F	Covering EVENT_MEMORY_FAILURE in the Handler availability table	11.1.0
SCP-68	SCP(15)000123	11.0.0	138		F	Covering EVENT_EVENT_ACCESS_TECHNOLOGY_CHANGE_MULTIPLE in the Handler availability table	11.1.0
SCP-65	SCP(14)000209	11.1.0	133		D	Update of Java Card™ reference	12.0.0
SCP-65	SCP(14)000210	11.1.0	134		C	Update of TerminalProfile class	12.0.0
SCP-66	SCP(14)000281	11.1.0	135		D	Correct description of getPollInterval method	12.0.0
SCP-67	SCP(15)000046	11.1.0	136		C	Update of ToolkitConstant Interface	12.0.0
SCP-68	SCP(15)000124	11.1.0	139		B	Support of Poll Interval Negotiation	12.0.0
SCP-69	SCP(15)000174	12.0.0	140		B	Supported Radio Access Technologies in PROVIDE LOCAL INFORMATION support in the API	13.0.0
SCP-71	SCP(15)000269	13.0.0	141		B	Add support for ENVELOPE (TERMINAL APPLICATIONS)	13.1.0
SCP-72	SCP(16)000018	13.0.0	142		B	Description of CAT ACTIVATE as a system command	13.1.0
SCP-73	SCP(16)000088	13.0.0	143		F	Update of references to GlobalPlatform specifications	13.1.0
SCP-73	SCP(16)000089	13.0.0	144		B	TERMINAL PROFILE eUICC Profile Switch constant addition	13.1.0
SCP-76	SCP(16)00234	13.0.0	146		F	Trigger *EVENT_FIRST_COMMAND_AFTER_ATR* event after profile change	13.1.0
SCP-79	SCP(17)000085	13.0.0	150	1	F	Correction in Terminal Profile Class	13.1.0
SCP-79	SCP(17)000086r1	13.0.0	151	1	F	Correction in Terminal Profile Class	13.1.0
SCP-75	SCP(16)000188	13.0.0.	145		B	TERMINAL PROFILE constant addition for eUICC Profile Operation	14.0.0
SCP-76	SCP(16)000235r1	13.0.0	147	1	B	Suspend Resume API	14.0.0

Meeting	Plenary Tdoc	VERS	CR	REV	CAT	SUBJECT	Resulting Version
SCP-76	SCP(17)000019r1	13.0.0	148	1	B	TerminalProfile constant addition for GET INPUT with Variable Time out	14.0.0
SCP-78	SCP(17)000050	13.0.0.	149		B	Suspend Resume Utility API	14.0.0
SCP-86	SCP(19)000020r3	12.0.0	155	1	D	Clarification of the Negotiation of Poll Interval behaviour	12.1.0
SCP-86	SCPTEC(19)000053r1	13.1.0	156	1	A	Clarification of the Negotiation of Poll Interval behaviour	13.2.0
SCP-86	SCPTEC(19)000054r1	14.0.0	157		A	Clarification of the Negotiation of Poll Interval behaviour	14.1.0

History

Document history		
V14.0.0	January 2019	Publication
V14.1.0	March 2019	Publication