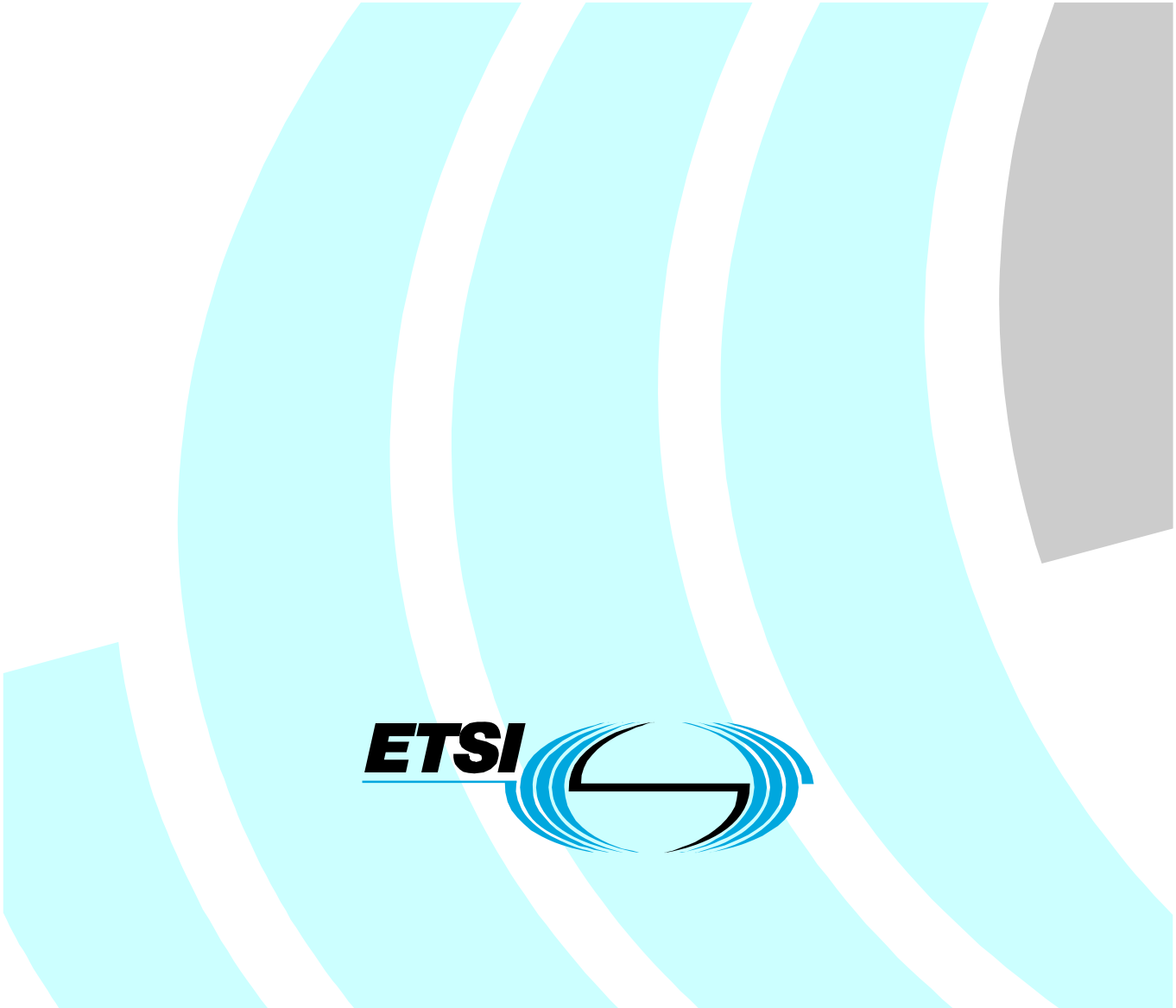
ETSI TS 102 241 V7.9.0 (2008-06)

Technical Specification

Smart Cards;
UICC Application Programming Interface (UICC API)
for Java Card™
(Release 7)



Reference
RTS/SCP-T0310R9

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 Technical Committee Smart Card Platform (SCP).

The present document details the stage 1 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 EP SPC 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.

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™ 2.2.2 specification [2], [3] and [4].

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

2 References

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.
- Non-specific reference may be made only to a complete document or a part thereof and only in the following cases:
  - if it is accepted that it will be possible to use all future changes of the referenced document for the purposes of the referring document;
  - for informative references.

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

For online referenced documents, information sufficient to identify and locate the source shall be provided. Preferably, the primary source of the referenced document should be cited, in order to ensure traceability. Furthermore, the reference should, as far as possible, remain valid for the expected life of the document. The reference shall include the method of access to the referenced document and the full network address, with the same punctuation and use of upper case and lower case letters.

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

2.1 Normative references

The following referenced documents are indispensable for the application of the present document. For dated references, only the edition cited applies. For non-specific references, the latest edition of the referenced document (including any amendments) applies.

[1] Void.

[2] Sun Microsystems Java Card™ Specification: "Java Card™ 2.2.2 Application Programming Interface".

[3] Sun Microsystems Java Card™ Specification: "Java Card™ 2.2.2 Runtime Environment (JCRE) Specification".

[4] Sun Microsystems Java Card™ Specification: "Java Card™ 2.2.2 Virtual Machine Specification".

NOTE: SUN Java Card Specifications can be downloaded at http://java.sun.com/products/javacard/specs.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: "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 (TM)".

2.2 Informative references

The following referenced documents are not essential to the use of the present document but they assist the user with regard to a particular subject area**.** For non-specific references, the latest version of the referenced document (including any amendments) applies.

Not applicable.

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:

ADF	Application Dedicated File
AID	Application IDentifier
APDU	Application Protocol Data Unit
API	Application Programming Interface
DF	Dedicated File (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)
TLV	Tag Length Value

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™ 2.2.2 API" [2], the Runtime Environment is an extension of the "Java Card™ 2.2.2 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 [2], [3] and [4]:

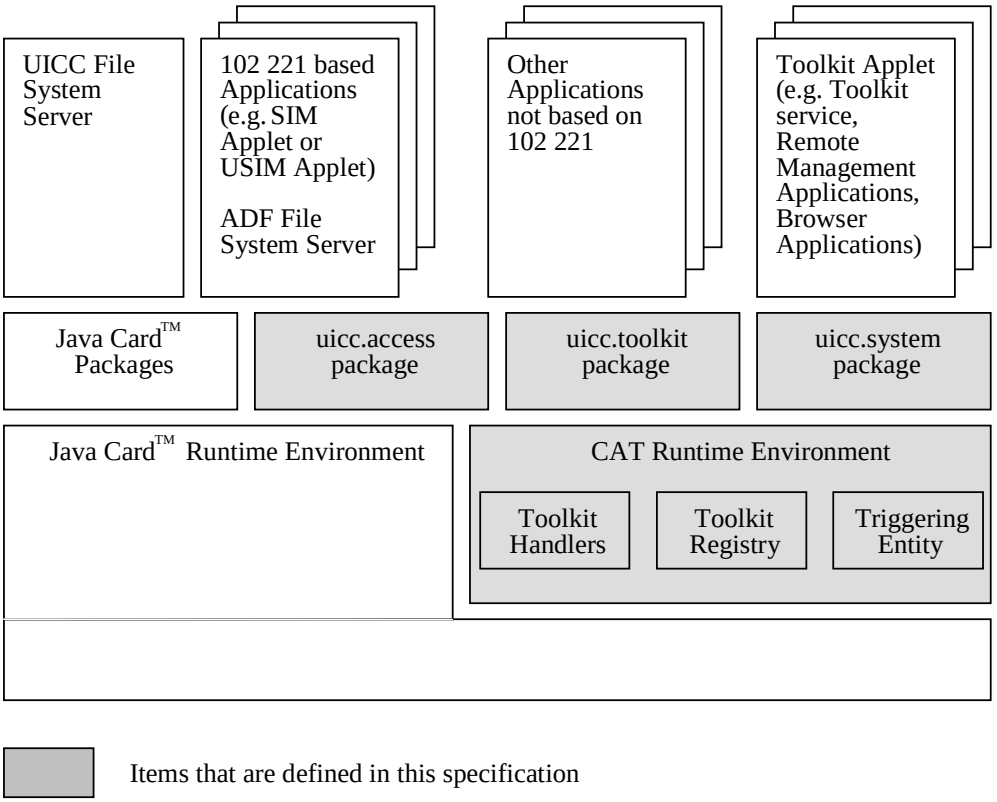


Figure 1: UICC Java Card™ architecture

Java Card™ Runtime Environment: this is specified in "Java Card™ 2.2.2 Runtime Environment (JCRE) Specification" [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.

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 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 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 "Java Card™ 2.2.2 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* 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™ 2.2.2 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 objects

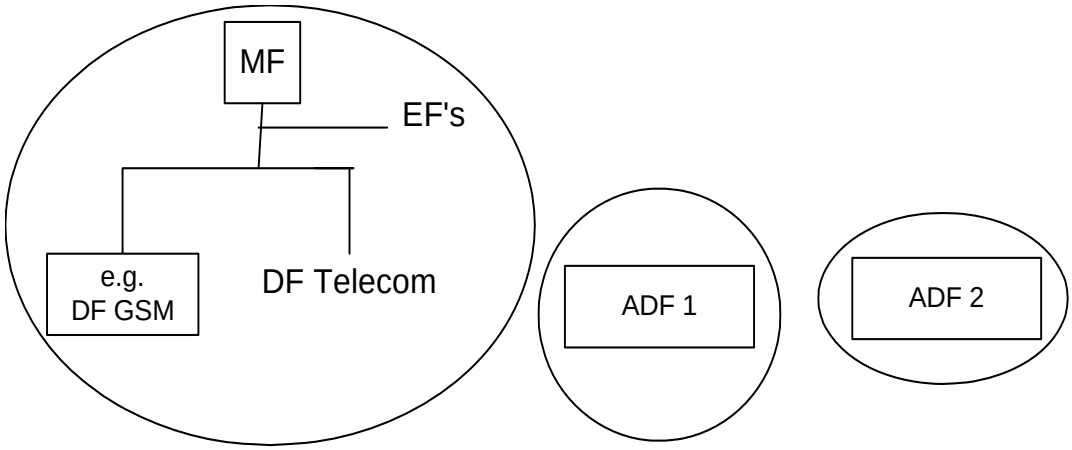


Figure 2: Logical structure of FileView

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 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 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 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].

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 "Java Card™ 2.2.1 Virtual Machine Specification" [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 TS 102 221 [6].
- SET DATA as defined in TS 102 221 [6].

6 Toolkit API and CAT Runtime Environment

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

6.1 Applet triggering

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.

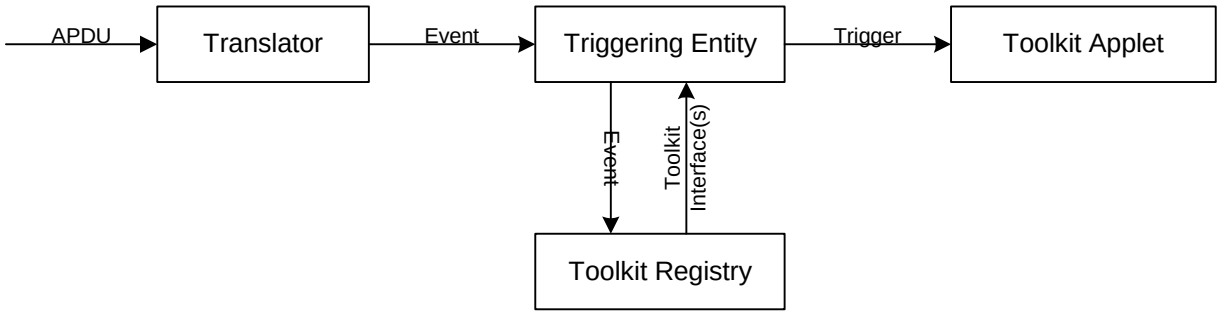


Figure 3: Toolkit applet triggering diagram

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 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). 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.

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
RFU	31 to 120
Reserved by 3GPP	121
Reserved by 3GPP	122
EVENT_PROACTIVE_HANDLER_AVAILABLE	123
EVENT_EXTERNAL_FILE_UPDATE	124
RFU	125
EVENT_APPLICATION_DESELECT	126
EVENT_FIRST_COMMAND_AFTER_ATR	127
RFU	128 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 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 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 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 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_ACCESS_TECHNOLOGY_CHANGE

EVENT_EVENT_DOWNLOAD_DISPLAY_PARAMETER_CHANGED

EVENT_EVENT_DOWNLOAD_NETWORK_SEARCH_MODE_CHANGE

EVENT_EVENT_DOWNLOAD_BROWSING_STATUS

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

EVENT_EVENT_DOWNLOAD_LOCAL_CONNECTION

Upon reception of an ENVELOPE (DOWNLOAD LOCAL CONNECTION) APDU as defined in 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 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 effect 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 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 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 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

Upon reception of the first APDU after the ATR and before the Status Word of the processed command 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 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 TS 101 220 [5]. 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.

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: 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 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 TS 102 223 [7];
- File List, as defined in 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 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 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.

6.3 Registration

A Toolkit applet shall register to the JCRE as specified in "Java Card™ 2.2.2 Runtime Environment (JCRE) Specification" [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 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 "Java Card™ 2.2.2 Runtime Environment (JCRE) Specification" [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.

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 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 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.

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
_DISPLAY_PARAMETER_CHANGED	Y	Y	N	n/n
_NETWORK_SEARCH_MODE_CHANGE	Y	Y	N	n/n
_BROWSING_STATUS	Y	Y	N	n/n
_LOCAL_CONNECTION	Y	Y	N	1/8 (per service identifier) (see note 2)
_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

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

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 "Java Card™ 2.2.2 (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.
  - 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

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 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 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 (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.

7 Toolkit applet

7.1 Applet loading

The UICC API card shall be compliant to the "Java Card™ 2.2.2 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™ 2.2.1 Application Programming Interface Specification" [2] and "Java Card™ 2.2.2 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™ 2.2.2 Runtime Environment Specification" [3] shall be used to delete the content from the UICC. The object deletion mechanism defined optional in "Java Card™ 2.2.2 Application Programming Interface Specification" [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 TS 102 221 [6].

8 UICC and ADF File System Administration API

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

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_102241v070900p0.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_102241v070900p0.zip, which accompanies the present document.

NOTE: See the "Java Card™ 2.2.2 Virtual Machine Specification" [4].

Annex C (normative): UICC API package version management

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

Table C.1

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	1.0	A0 00 00 00 09 00 05 FF FF FF FF 89 12 00 00 00	1.0
	A0 00 00 00 09 00 05 FF FF FF FF 89 11 00 00 00	1.1	A0 00 00 00 09 00 05 FF FF FF FF 89 12 00 00 00	1.1
	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	1.2

Table C.2

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	1.0

Table C.3

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

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

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.

Annex D (informative): Menu order example

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 change requests approved for the present document since the first version was approved.

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 parameters.	
	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	49		F	Clarification for non-specific references	6.6.0
			50		F	Terminal Profile update to the latest changes in 102 223	
			52		F	Definition of TAR_NOT_DEFINED for ToolkitException	
			53		F	Clarification for Exception in case capacity is negative	
			54		F	Clarification for the *EVENT_EXTERNAL_FILE_UPDATE*	
			55		F	Terminal Profile update for text attribute features	
SCP-19	SCP-040432	6.6.0	51		D	Clarification in description of AdminFileView	7.0.0
SCP-20	SCP-050019	7.0.0	57		A	Corrections in documentation of Java methods for file event registration	7.1.0
			59		A	Access rights clarification for FileView and AdminFileView	
			61		A	Correction of SET UP EVENT LIST system command behaviour	
			63		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 TS 102 221 and TS 102 222	
SCP-27	SCP-060445	7.5.0	101		A	Correction of the release for references	7.6.0
	SCP-060476		97	1	F	Reserve a short identifier for a 3GPP event defined in 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	
			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).	7.8.0
TS 102 241 v7.8.0 was withdrawn (decision made at SCP #35)							
SCP-30bis	SCP-070191	7.7.0	104	2	B	Addition of a method for concurrent card application toolkit sessions	7.9.0
SCP-33	SCP-070419	7.7.0	110		B	Introduction of EVENT_REMOTE_FILE_UPDATE	
			111		F	Modification of CAT Runtime Environment behaviour in case of CLOSE CHANNEL command for UICC Server	

Meeting	Plenary Tdoc	VERS	CR	REV	CAT	SUBJECT	Resulting Version
						Mode in mode "TCP in LISTEN state"	
			112		D	Editorial Correction in Method uicc.system.HandlerBuilder.buildTLVHandler()	

History

Document history		
V7.0.0	December 2004	Publication
V7.1.0	April 2005	Publication
V7.2.0	October 2005	Publication
V7.3.0	January 2006	Publication
V7.4.0	May 2006	Publication
V7.5.0	July 2006	Publication (withdrawn)
V7.5.1	September 2006	Publication
V7.6.0	October 2006	Publication
V7.7.0	February 2007	Publication
V7.9.0	June 2008	Publication