ETSI TS 102 241 V16.1.0 (2020-02)

Smart Cards;
UICC Application Programming Interface (UICC API)
for Java Card™
(Release 16)

Reference
RTS/SCP-T0310vg10

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

Important notice

The present document can be downloaded from:
http://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the prevailing version of an ETSI deliverable is the one made publicly available in PDF format at www.etsi.org/deliver.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

Copyright Notification

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 2 description of the "Application Programming Interface and Loader Requirements" [11] internal to the UICC.

This stage 2 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 https://www.oracle.com/technetwork/java/embedded/javacard/downloads/default-1970005.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)".

[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 Contactless 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: 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 (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].

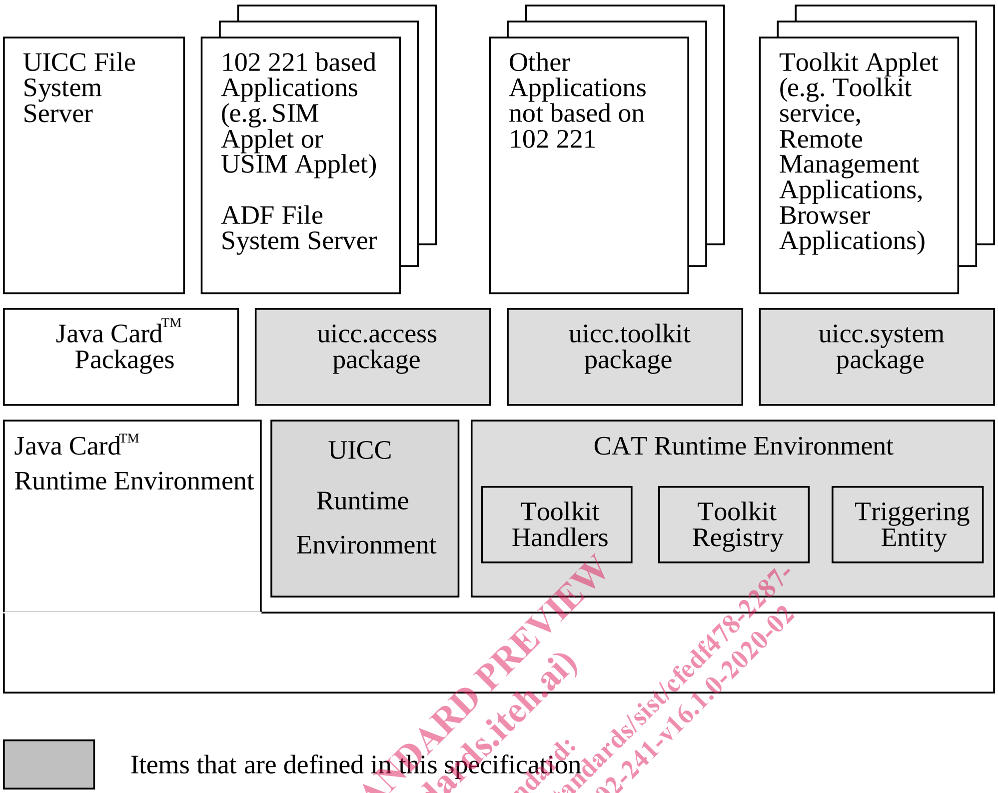

Figure 1: UICC Java Card™ architecture

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

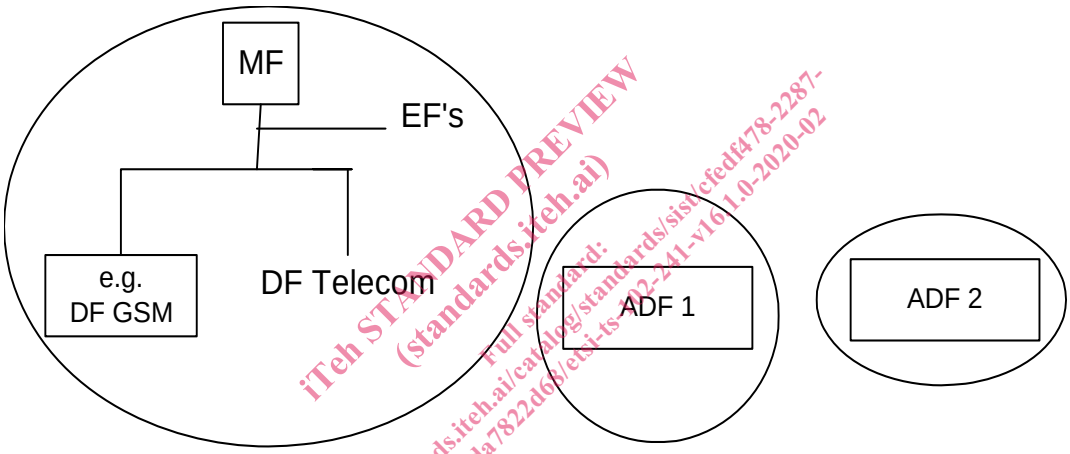

Figure 2: Logical structure of FileView

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.

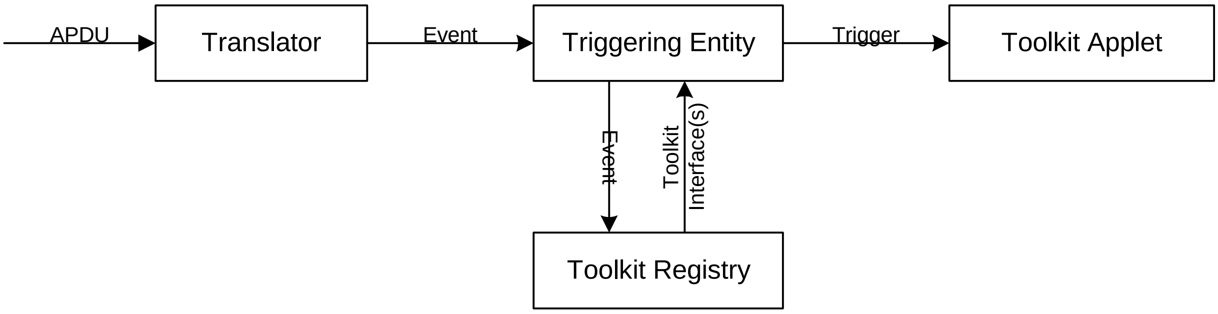

Figure 3: Toolkit applet triggering diagram

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.