

ETSI TS 102 705 V11.0.0 (2013-09)



Technical Specification

**Smart Cards;
UICC Application Programming Interface for Java Card™
for Contactless Applications
(Release 11)**



Reference

RTS/SCP-THCIAPIVB00

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

Individual copies of the present document can be downloaded from:

<http://www.etsi.org>

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

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

<http://portal.etsi.org/tb/status/status.asp>

If you find errors in the present document, please send your comment to one of the following services:

http://portal.etsi.org/chaicor/ETSI_support.asp

Copyright Notification

No part may be reproduced except as authorized by written permission.
The copyright and the foregoing restriction extend to reproduction in all media.

© European Telecommunications Standards Institute 2013.
All rights reserved.

DECT™, PLUGTESTS™, UMTS™ and the ETSI logo are Trade Marks of ETSI registered for the benefit of its Members.
3GPP™ and LTE™ are Trade Marks of ETSI registered for the benefit of its Members and
of the 3GPP Organizational Partners.
GSM® and the GSM logo are Trade Marks registered and owned by the GSM Association.

Contents

Intellectual Property Rights	4
Foreword.....	4
1 Scope	5
2 References	5
2.1 Normative references	5
2.2 Informative references.....	6
3 Definitions and abbreviations.....	6
3.1 Definitions.....	6
3.2 Abbreviations	6
4 Description	7
4.1 Architecture	7
4.2 Card Emulation Mode	9
4.3 Reader Mode	9
4.4 Connectivity Service	10
5 Interaction with Proactive Functionality	10
6 Java Card Resource Handling	11
Annex A (normative): Java Card™ Platform HCI API for the UICC	12
Annex B (normative): Java Card™ Platform HCI API for the UICC identifiers	13
Annex C (normative): HCI API package version management.....	14
Annex D (informative): Change history	15
History	16

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://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.

Foreword

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

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 modifies the contents of the present document, it will then be republished by ETSI with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
 - 0 early working draft;
 - 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 describes the UICC Application Programming Interface for Java Card™ for contactless Applications. Its purpose is to provide access for a contactless Applet to the services provided by the HCI protocol defined in [4] for the communication via the CLF. In the scope of the present document contactless means support for the RF Technologies referenced by the HCI specification [4]. Low level functionality to manage gates and pipes as defined in the HCI specification [4] is not in the scope of the present document. Registration of contactless parameters and management of contactless Applets in card emulation mode is defined in "GlobalPlatform Card Specification Amendment C" [8]. Related APIs are provided in "Java Card API and Export File for Card Specification v2.2.1 (org.globalplatform) v1.5" [12] and "Java Card Contactless API and Export File for Card Specification v2.2.1 (org.globalplatform.contactless)" [13].

2 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 reference 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.

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

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 necessary for the application of the present document.

- [1] ISO/IEC 7816-3 (2006): "Identification cards - Integrated circuit cards - Part 3: Cards with contacts - Electrical interface and transmission protocols".
- [2] ETSI TS 102 221: "Smart Cards; UICC-Terminal interface; Physical and logical characteristics".
- [3] ETSI TS 101 220: "Smart Cards; ETSI numbering system for telecommunication application providers".
- [4] ETSI TS 102 622: "Smart Cards; UICC - Contactless Front-end (CLF) Interface; Host Controller Interface (HCI)".
- [5] ETSI TS 102 241: "Smart Cards; UICC Application Programming Interface (UICC API) for Java Card (TM)".
- [6] ETSI TS 102 223: "Smart Cards; Card Application Toolkit (CAT)".
- [7] ETSI TS 102 226: "Smart Cards; Remote APDU structure for UICC based applications".
- [8] GlobalPlatform: "GlobalPlatform Card Specification Version 2.2, Amendment C: Contactless Services" Version 1.0.

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

- [9] Sun Microsystems "Application Programming Interface, Java Card™ Platform, 3.0.1 Classic Edition".
- [10] Sun Microsystems "Runtime Environment Specification, Java Card™ Platform, 3.0.1 Classic Edition".

- [11] Sun Microsystems "Virtual Machine Specification Java Card™ Platform, 3.0.1 Classic Edition".
- NOTE: SUN Java Card Specifications can be downloaded at <http://java.sun.com/products/javacard/specs.html>.
- [12] GlobalPlatform: "Java Card API and Export File for Card Specification v2.2.1 (org.globalplatform)" v1.5.
- [13] GlobalPlatform: "Java Card Contactless API and Export File for Card Specification v2.2.1 (org.globalplatform.contactless)" v1.1.
- [14] ETSI TS 102 613: "Smart Cards; UICC - Contactless Front-end (CLF) Interface, Part 1: Physical and data link layer characteristics".

2.2 Informative references

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 Definitions and abbreviations

3.1 Definitions

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

contactless mode: used as a generic term for "Card Emulation Mode" and "Reader Mode"

contactless state: corresponds to the logical state of the contactless framework

HCP message: message as specified in TS 102 622 [4]

NOTE: An HCP message can be of type "command", "event" or "response to a command".

RF Technology: radio frequency technology supported by the HCI (TS 102 622 [4]) protocol specification

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

APDU Application Protocol Data Unit

NOTE: According to ISO/IEC 7816-3 [1].

API Application Programming Interface

CLF Contactless Front-end

NOTE: According to TS 102 622 [4].

CRS Contactless Registry Service

HCI Host Controller Interface

NOTE: According to TS 102 622 [4].

HCP Host Controller Protocol

NOTE: According to TS 102 622 [4].

RF Radio Frequency

SWP Single Wire Protocol

NOTE: According to TS 102 613 [14].

4 Description

4.1 Architecture

The present document describes an API and a Contactless Framework that enables Java Card™ Platform based Applets, defined in [9], [10] and [11], to send and receive messages using the HCI protocol as specified in TS 102 622 [4] and to act as contactless Applets. The Contactless Framework shall support card emulation mode and reader mode as specified in the HCI protocol (TS 102 622 [4]) specification.

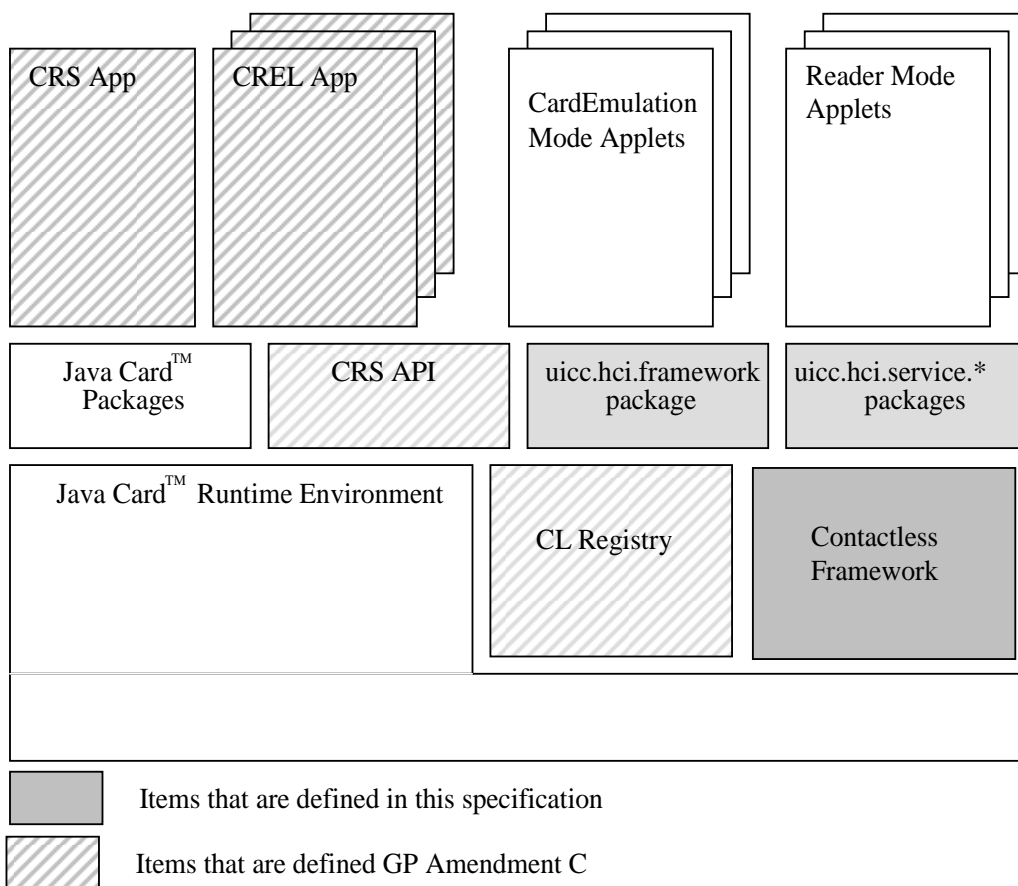


Figure 1

The functionality of the Contactless Framework and the configuration of contactless parameters and the management of contactless Applets in card emulation mode are based on the functionality provided by the Contactless Registry Service (CRS), the related APIs, the CRS Application and other features and concepts which are defined in the "GlobalPlatform Amendment C" [8] and the related APIs "Java Card API and Export File for Card Specification v2.2.1 (org.globalplatform) v1.5" [12] and "Java Card Contactless API and Export File for Card Specification v2.2.1 (org.globalplatform.contactless)" [13].

The API is event driven and based on the Observer/Listener pattern. Every HCI service is encapsulated by a dedicated Service interface. These Service interfaces shall allow the registration of Listener Interfaces and the activation of events. The Listener Interfaces shall be implemented by Java objects to receive HCI messages and events in the *onCallback* method. The Registration of Listener Interfaces and activation of events shall be persistent.

An *HCIMessage* object shall encapsulates one HCP message according to the HCI protocol as specified in TS 102 622 [4]. HCI message for the different contactless modes shall be identified by different types of interfaces. It is not guaranteed that any Applet originated HCI messages are sent before the completion of the execution of the current Applet. The Contactless Framework sends the Applet originated HCI messages in the same order as they are submitted by the Applet.

NOTE 1: The Contactless Framework may not have enough resources to send several HCI messages submitted during the same *onCallback* method execution. The Applet should be aware of this limitation (e.g. use suitable error handling strategy, or send only one HCI message in the *onCallback* method at a time).

Any *onCallback()* method of a Listener interface shall not be invoked again while another *onCallback()* method is still being executed. The Contactless Framework shall be able to receive one or more HCI messages while waiting for a response related to a command originated by the Applet (e.g. processing a request for parameters) especially for the *EVT_FIELD_OFF* case.

The HCI event *EVT_FIELD_OFF* shall be buffered and sent by the Contactless Framework as soon as the Contactless Framework becomes the current context.

All other HCI messages shall be delivered to the Applet instance in the same order as they were received by the Contactless Framework.

Contactless State is the logical state of the Contactless Framework it can take the value enabled and disabled. It refers to the "contactless functionality in the UICC" as used in TS 102 223 [6]

This state can be changed with the mechanisms defined in TS 102 223 [6], and by the method *setCommunicationInterface()* API method of "GlobalPlatform Amendment C" [8].

The Contactless State applies only to the Card Emulation Mode and the Reader Mode, and it does not apply to the Connectivity service.

When the Contactless State is disabled, the Contactless Framework shall throw an *HCIException* with reason code *HCI_CURRENTLY_DISABLED* when an Applet invokes a method which requires that the Contactless State is enabled.

When the Contactless State is enabled and the state of the SWP [14] interface is *DEACTIVATED* and when the Contactless Framework needs to send data over the SWP [14] interface then it shall send the proactive command *ACTIVATE* defined in TS 102 223 [6] if supported by the terminal.

NOTE 2: An Applet may use the method *HCIDevice.isHCIServiceAvailable()* to check if the Contactless Framework supports sending the *ACTIVATE* command on pre Rel-11 implementations.

The API is split into two parts. One is a generic framework that provides a factory class to retrieve the different Service instances that are provided by the HCI implementation, and that allows discovery of whether the UICC is inserted into a HCI network. The second part of the API implements the Services that are defined for the HCI protocol, card emulation mode, reader mode and connectivity service.

As stated in TS 102 622 [4], a reset of the underlying data link layer shall be transparent to the application layer if the data link layer recovers without any loss of data. In the case of a data loss an error shall be reported back to the Application which has initiated the data transmission. In case of a communication in CardEmulation Mode where a command from an external reader cannot be forwarded to an Application due to a data link layer problem the Contactless Framework shall not take any action but wait for the reader to resolve the problem.

4.2 Card Emulation Mode

In card emulation mode there exist two exclusive ways to exchange messages over the HCP [4]. The first is based on APDUs provided to the Applet through its *process()* method as specified in "Application Programming Interface, Java Card™ Platform, 3.0 Classic Edition" [9]. The second is made available by the package *uicc.hci.services.cardemulation* defined in this specification.

The *uicc.hci.services.cardemulation* package shall provide the communication technologies for the card emulation mode defined by the HCP as specified in [4]. The Contactless Framework shall bind the services defined in the *uicc.hci.services.cardemulation* package to the underlying HCI resources (e.g. gates and pipes) defined in the HCI architecture as specified in [4]. The parameters to be used by the HCI layer may be provided to the framework as defined in "GlobalPlatform Amendment C" [8].

For the API defined in this specification the card emulation capability shall be provided to Applets through a service interface implemented by the Contactless Framework. Applet instances shall receive *CardEmulationMessages* after the registration of a *CardEmulationListener* interface to a *CardEmulationService* only if the *EVENT_ON_SEND_DATA* is activated for the Applet instance. If the *EVENT_ON_SEND_DATA* is deactivated for the Applet instance and an APDU is received via the *EVT_SEND_DATA*, the *javacard.framework.APDU* class and the *process()* method of the Applet instance shall be invoked.

It shall not be possible to switch between the usage of the *CardEmulationListener* interface and the invocation through the *process()* method within a contactless application session, i.e. not before the Applet has been deselected and selected again. Applets communicating through the *process()* method shall also be able to use the API services defined in this specification which do not require a *CardEmulationListener* registration (e.g. requesting the power mode or connectivity service).

If the current application was selected through a SELECT by DF name, the Contactless Framework shall handle an application session termination according to TS 102 221 [2] independent of the interface used for APDU exchange.

Applet selection and deselection shall be performed by the Contactless Framework according to the rules defined in the "Java Card™ Runtime Environment Specification, 3.0.1 Classic Edition" [10] and in "GlobalPlatform Amendment C" [8].

The *select()* method of the Applet instance shall always be invoked for an Applet selection according to the rules given in "Java Card™ Runtime Environment Specification, 3.0.1 Classic Edition" [10].

In case the Applet instance has registered the *CardEmulationListener* and has activated the *EVENT_ON_SEND_DATA* the *process()* method of this Applet instance shall not be invoked during the selection. The *CardEmulationListener.onCallback* method shall be called by the Contactless Framework. The HCP message that resulted in the selection of this Applet according to the rules defined in "GlobalPlatform Amendment C" [8] shall be provided by the *CardEmulationMessage*.

If the HCI event *EVT_FIELD_OFF* or *EVT_CARD_DEACTIVATED* defined by the HCP specified in TS 102 622 [4] is received by the Contactless Framework and the UICC is still powered, the Applet instance shall be deselected according to "GlobalPlatform Amendment C" [8] by invocation of the *deselect()* method.

When the HCI event *EVT_FIELD_OFF* is received and if the Applet instance has activated this event the Contactless Framework shall raise an *EVENT_FIELD_OFF* before the invocation of the *deselect()* method of the Applet instance.

After the deselection of the Applet instance, it shall not be invoked by any other event defined in the interface *CardEmulationListener* until the Applet instance is selected again.

4.3 Reader Mode

The functionality to support the reader mode is provided in the package *uicc.hci.services.reader*. In reader mode the communication technologies defined by the contactless platform for reader mode [4] are supported. The Contactless Framework shall bind the services defined in *uicc.hci.services.reader* to the corresponding resources (e.g. gates and pipes) defined by the contactless platform for reader mode [4].

An Applet has to be in the selectable state (according to the Java Card™ specification [9], [10] and [11]) to act as a contactless Applet in reader mode.

Reader mode Applets shall follow the extended lifecycle model that is defined in "GlobalPlatform Amendment C" [8] for contactless Applets in card emulation mode (i.e. following Application Availability States and the related transition rules).

Per RF technology there shall be only one reader mode Applet in the state ACTIVATED (according to "GlobalPlatform Amendment C" [8]) at any time.

The installation parameters for contactless reader mode applications are specified in TS 102 226 [7].

When the state of a reader mode Applet changes to lifecycle ACTIVATED (according to "GlobalPlatform Amendment C" [8]) the Contactless Framework shall ensure that the HCI gates and pipes are setup for the RF technologies that are supported by the reader mode Applet.

To be able to receive and send messages over the contactless interface in reader mode the Applet shall activate the *ReaderListener.EVENT_TARGET_DISCOVERED*. An Applet shall only be able to activate this event or to use the *restartReadermodeProcedure* method if it is in lifecycle state ACTIVATED. To release the CLF control at the end of a transaction an Applet shall deactivate the *ReaderListener.EVENT_TARGET_DISCOVERED*.

When an Applet lifecycle state changes from ACTIVATED to DEACTIVATED the Contactless Framework shall enforce that the *ReaderListener.EVENT_TARGET_DISCOVERED* is deactivated.

The Contactless Framework shall request the reader mode control on the CLF by sending the HCI events *EVT_READER_REQUESTED* and *EVT_END_OPERATION* according to the state of the reader mode Applet. The *EVT_READER_REQUESTED* shall be sent by the Contactless Framework if an Applet instance activates the event *ReaderListener.EVENT_TARGET_DISCOVERED* and no other Applet instance has the event activated, i.e. it shall not be sent if the Contactless Framework has earlier sent an *EVT_READER_REQUESTED* due to the request from another Applet instance, which was not yet ended by an *EVT_END_OPERATION*. The HCI event *EVT_END_OPERATION* shall be sent to the CLF when an Applet instance or the Contactless Framework deactivates the event *ReaderListener.EVENT_TARGET_DISCOVERED*. The Contactless Framework shall resend the *EVT_READER_REQUESTED* to the CLF if another Applet instance exists with the *ReaderListener.EVENT_TARGET_DISCOVERED* event activated.

The Contactless Framework shall inform the Applet instance which has activated the *ReaderListener.EVENT_TARGET_DISCOVERED* when a target is discovered on one of the RF technologies the Applet instance is registered to with its installation parameters as specified in TS 102 226 [7].

The Contactless Framework shall ensure that the *ReaderListener.EVENT_TARGET_DISCOVERED* is deactivated for all Applets when access to the interface is disabled on the UICC level.

4.4 Connectivity Service

The functionality to support the connectivity mechanisms specified in the HCI specification [4] is provided in the package *uicc.hci.services.connectivity*. The Contactless Framework shall bind the services defined in *uicc.hci.services.connectivity* to the corresponding resources (e.g. gates and pipes) specified in the HCI specification [4] for connectivity.

The Contactless Framework shall only accept the request to send the HCI event *EVT_CONNECTIVITY* or *EVT_TRANSACTION* specified by the HCP [4] when initiated by an Applet instance calling one of the *ConnectivityService* interface methods, if the Applet is the selected Applet in card emulation mode or is in the state ACTIVATED (according to "GlobalPlatform Amendment C" [8]) for the reader mode.

5 Interaction with Proactive Functionality

The *ProactiveHandler* defined in TS 102 241 [5] shall not be available when the contactless Applet is invoked with the callback methods defined in this specification, or when the Applet is invoked with the *process()* method of the Applet class defined in Application Programming Interface, Java Card™ Platform [9] (in card emulation mode). If the Applet wants to use proactive functionality it shall use the Connectivity Service defined above to send an HCI event *EVT_CONNECTIVITY* to the terminal, register for *EVENT_EVENT_DOWNLOAD_HCI_CONNECTIVITY* and return. All the proactive functionality of the UICC API defined in TS 102 241 [5] is then available to the Applet when that Applet instance is triggered with the *processToolkit()* method defined in TS 102 241 [5].

6 Java Card Resource Handling

The Runtime Environment invokes an Applet by calling the *onCallback()* method of the respective contactless Listener Interface. As a consequence all the rules defined in "Java Card™ Platform 3.0.1 Classic Edition, Runtime Environment Specification" [10] apply (e.g. access to CLEAR_ON_DESELECT transient objects, context switch, multi selectable).

When the *onCallback()* method of a Listener Interface is invoked, no transaction shall be in progress.

The context as defined in the Java Card™ specification [9], [10] and [11] shall be set to the context of the Applet which implements the *onCallback()* method. The previous context (context of the caller) shall be the context of the Contactless Framework.

Upon return from the *onCallback()* method a pending transaction shall be aborted.

Annex A (normative): Java Card™ Platform HCI API for the UICC

The source files for the UICC Application Programming Interface for Java Card™ for contactless Applets (102705_Annex_A_Java.zip and 102705_Annex_A-HTML.zip) are contained in ts_102705v110000p0.zip, which accompanies the present document.

Annex B (normative): Java Card TM Platform HCI API for the UICC identifiers

The export files for the uicc.hci.* package (102705_Annex_B_Export_Files.zip) are contained in ts_102705v110000p0.zip, which accompanies the present document.

Annex C (normative): HCI API package version management

Table C.1 describes the relationship between each TS 102 705 specification version and its HCI API packages AID and Major, Minor versions defined in the export files.

Table C.1

TS 102 705	uicc.hci.framework package	
	AID	Major, Minor
11.0.0	A0 00 00 00 09 00 05 FF FF FF FF 89 16 01 00 00	1.1

Table C.2

TS 102 705	uicc.hci.services.cardemulation package	
	AID	Major, Minor
11.0.0	A0 00 00 00 09 00 05 FF FF FF FF 89 16 02 01 00	1.0

Table C.3

TS 102 705	uicc.hci.services.connectivity package	
	AID	Major, Minor
11.0.0	A0 00 00 00 09 00 05 FF FF FF FF 89 16 02 02 00	1.0

Table C.4

TS 102 705	uicc.hci.services.readermode package	
	AID	Major, Minor
11.0.0	A0 00 00 00 09 00 05 FF FF FF FF 89 16 02 03 00	1.0

The package AID coding is defined in TS 101 220 [3]. The HCI API packages' AIDs 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): Change history

Meeting	Document	CR	Rev	Cat	Title	Resulting Version
SCP#45					First publication of the specification	9.0.0
SCP#46	SCP(10)0286	001	1	F	Clarification about restartReaderModeProcedure method use	9.1.0
SCP#46	SCP(10)0287	002	1	F	Wrong Reference in HCIDevice.java related to UICC power mode	9.1.0
SCP#46	SCP(10)0288	003	-	F	Correction of wrong constant value and name	9.1.0
SCP#46	SCP(10)0289	004	-	F	Clarification of contactless framework behaviour when HCI event EVT_FIELD_OFF is received	9.1.0
SCP#46	SCP(10)0291	007	-	F	Addition of uses of HCI_CONDITIONS_NOT_SATISFIED in ConnectivityService	9.1.0
SCP#46	SCP(10)0292	008	-	F	Clarification of use of HCI_CURRENTLY_DISABLED reason code	9.1.0
SCP#46	SCP(10)0309	006	-	F	Handling of HCI data link layer reset by HCI Framework	9.1.0
SCP#47	SCP(11)0053	009	-	F	Behaviour of HCIDevice.getHCIService() method when Applet.register() method has not been invoked.	9.2.0
SCP#47	SCP(11)0055	010	-	F	Correction of prepareAndSendGetParameterCommmand() method name	9.2.0
SCP#47	SCP(11)0056	011	-	F	HCIMessage content after requestCallbackNotification()	9.2.0
SCP#47	SCP(11)0057	012	-	F	New exception for HCIService.activateEvent() method	9.2.0
SCP#47	SCP(11)0058	013	-	F	WRITE_EXCHANGE event for HCIService activation, deactivation and get status methods	9.2.0
SCP#47	SCP(11)0054	014	-	F	Range specification for event parameter in requestCallbackNotification()	9.2.0
SCP#47	SCP(11)0052r1	005	2	C	CR 102 705 R10 #005r2: Clarification about HCI messages sending order	10.0.0
SCP#48	SCP(11)0160	016	-	A	CR 102 705 R10 #016: Clarification of use of events EVENT_HCI_RECEPTION_FAILED and EVENT_HCI_TRANSMISSION_FAILED	10.0.0
SCP#52	SCP(11)0316	021		A	Clarification of HCIMessage.getReceiveBuffer(), getReceiveOffset() and getReceiveLength()	10.1.0
SCP#52	SCP(11)0317	022		A	Missing constant in HCIMessage corresponding to HCI response ANY_E_PIPE_NOT_OPENED	10.1.0
SCP#52	SCP(11)0318	023		A	Correction of several inaccuracies in HCIMessage.isHeading() and isComplete()	10.1.0
SCP#52	SCP(11)0319r1	024	1	A	Removal of requestCallbackNotification functionality	10.1.0
SCP#56	SCP(12)000217	034		A	Define the relationship between the access to the contactless interface and the install parameters	10.1.0
SCP#56	SCP(12)000218	035		A	Activation of events	10.1.0
SCP#56	SCP(12)000219	036		A	Clarification of service availability	10.1.0
SCP#56	SCP(12)000220	037		A	Correct usage of Listener Interfaces	10.1.0
SCP#54	SCP(12)000017	028		C	Deprecation of two constants for reading CLF registry	11.0.0
SCP#56	SCP(12)000158	029	3	C	Clarifications related to the use of the proactive command ACTIVATE	11.0.0

History

Document history		
V11.0.0	September 2013	Publication