ETSI TS 131 221 V11.0.0 (2012-11)

Universal Mobile Telecommunications System (UMTS);
LTE;
Contact Manager Application Programming Interface (API);
Contact Manager API for Java Card
(3GPP TS 31.221 version 11.0.0 Release 11)

Reference
RTS/TSGC-0631221vb00

Keywords
LTE,UMTS

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

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 3rd Generation Partnership Project (3GPP).

The present document may refer to technical specifications or reports using their 3GPP identities, UMTS identities or GSM identities. These should be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between GSM, UMTS, 3GPP and ETSI identities can be found under http://webapp.etsi.org/key/queryform.asp.

Contents

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG 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 TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG 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 internal interface characteristics of the Contact Manager for 3GPP UICC applications [2].

The internal interface between the Contact Manager Server application on the UICC and the Contact Manager Client application on the UICC enables Java Card™ platform based applets, defined in [3], [4] and [5], to invoke and register to the Contact Manager Server services. In particular, the Contact Manager Java Card™ based API provides methods to:

- Read/Update/Create/Delete contact(s) in the Contact Manager Server;
- Manage group of contacts in the Contact Manager Server;
- Search for a contact in the Contact Manager Server storage;
- manage the contacts structure;
- Register/Un-register the application to pre-defined events (e.g. application to be notified when contacts are modified in the Contact Manager Server).

This API allows to develop an application running together with a Contact Manager [2].

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document.*

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 31.220: "Characteristics of the Contact Manager for 3GPP UICC applications".

[3] Sun Microsystems "Application Programming Interface, Java Card™ Platform, 3.0.1 Classic Edition".

[4] Sun Microsystems "Runtime Environment Specification, Java Card™ Platform, 3.0.1 Classic Edition".

[5] 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

[6] 3GPP TS 31.130: "(U)SIM API for Java™ Card".

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

3 Definitions, symbols and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TR 21.905 [1] and TS 31.130 [6] apply. A term defined in TS 31.130 [6] takes precedence over the definition of the same term, if any, in TR 21.905 [1].

3.2 Symbols

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

|| Concatenation

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in TR 21.905 [1] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in TR 21.905 [1].

API Application Programming Interface

4 Contact manager internal interface characteristics

4.1 Reference model

The present section describes an API and a Contact Manager Runtime Environment that enables Java Card™ platform based applets, defined in [3], [4] and [5], to invoke and register to the Contact Manager Server services (e.g. retrieve/modify contact information). The Contact Manager Runtime Environment API is further described in annex A.

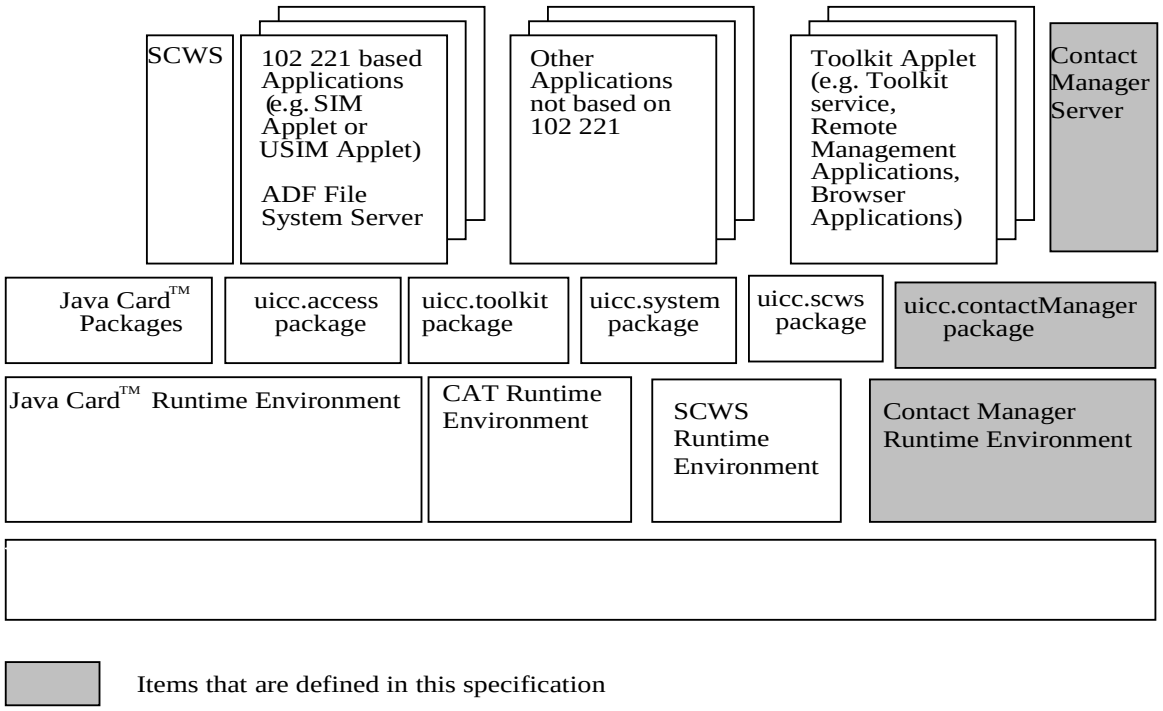

Figure 1: Internal interface reference model

Contact Manager Runtime Environment: An Extension to the Java Card™ platform described in [3][4][5] and the (U)SAT Runtime Environment described in TS 31.130 [6] to facilitate the communications between Applets and the Contact Manager Server.

Applet: 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" [4].

Event Listener List of the Contact Manager: this is handling all the registration information of the event listener of the Contact Manager Server services. It is provided as a JCRE entry point object defined in [5]. The registry is part of the Contact Manager Runtime Environment.

Contact Manager API: consists of the package *uicc.contactmanager*, provides the methods to register and unregister, to the Contact Manager events. It also provides methods to access the Contact Manager Server data. This API is an extension to the "(U)SAT API" defined in TS 31 130 [6].

Contact Manager Server: application on the UICC that enables Java Card™ platform based applets, defined in [3], [4] and [5], to invoke and register to its services. The invocation of the Contact Manager Server services is done through methods corresponding to the Contact Manager Runtime Environment interface.

4.2 Data model

The Contact Manager API gives the user application a view of the data stored by the Contact Manager Server. This view follows the data model below:

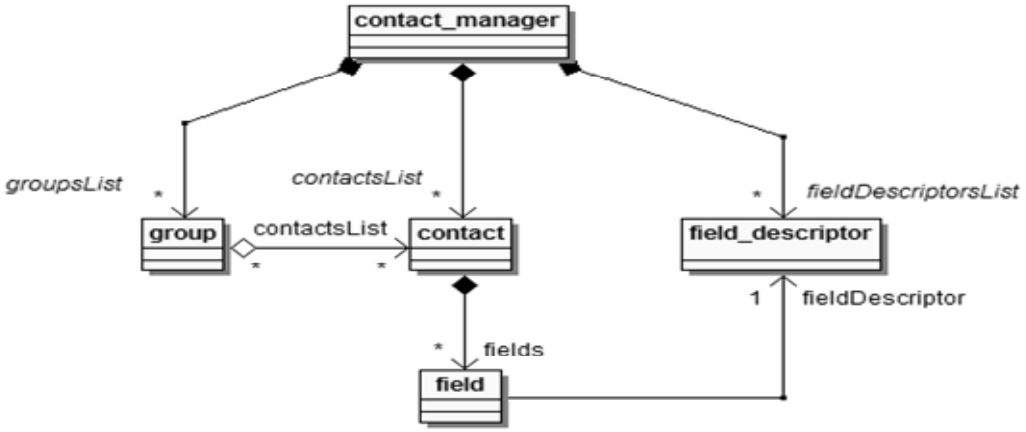

Figure 1: data model

The contact manager is composed of a collection of contacts. Each contact belongs to a unique contact manager.

The contact manager is also composed of a collection of field descriptors. Each field descriptor defines the properties of a field that all contacts belonging to the contact manager shall have. A contact field is described by:

- its type (e.g. Name, phone number, postal address,..) and
- its attributes (e.g. voice phone number).

Some fields such as a name or a postal address may be composed of several sub-fields. Each field descriptor belongs to a unique contact manager.

A contact is composed of a collection of fields. Each field belongs to a unique contact, and defines the value assigned to this contact for a unique field descriptor.

The contact manager is also composed of a collection of groups. Each group belongs to a unique contact manager, and is an aggregation of contacts that belong to the same contact manager. Each contact may belong to no group, one group, or multiple groups.

Each contact is identified by an identifier, which is unique among all contacts in the Contact Manager database.

Each field descriptor is identified by an identifier, which is unique among all field descriptors in the Contact Manager database.

Each group is identified by an identifier, which is unique among all groups in the Contact Manager database.

The contact identifiers, field descriptor identifiers and group identifiers are three distinct spaces. For example, a contact may have the same identifier as a group.

The following figure depicts an example of a Contact Manager Server data store.

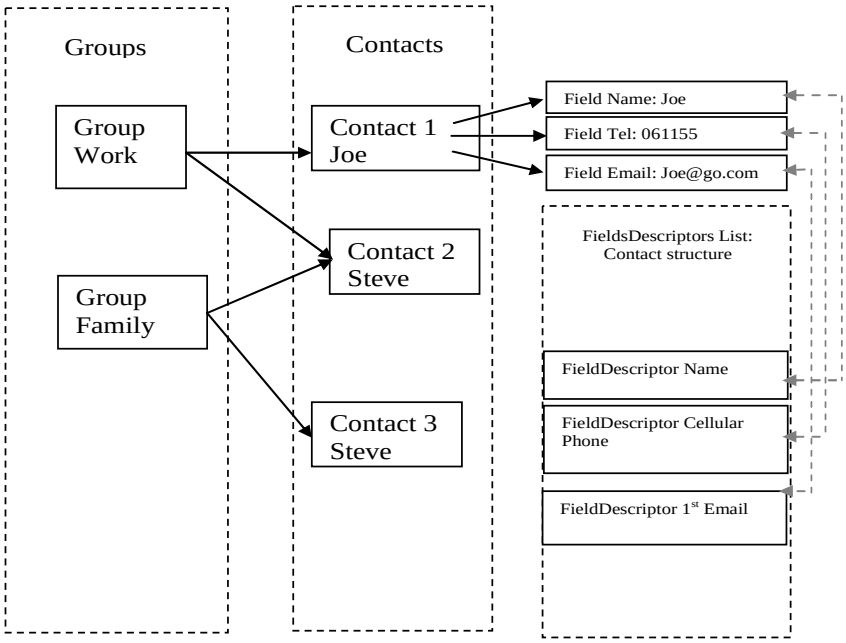

Figure 2: Example of a Contact Manager Server data store

4.3 Events registration and deregistration

4.3.1 Overview

Registration to and Deregistration from the Contact Manager Server is done through the register and unregister methods of the Contact Manager Runtime Environment interface.

To be notified, the client shall implement a *ContactEventListener interface* and register to the appropriate events. Then the Contact Manager Server will call the following methods of all registered clients when the corresponding event occurs:

- *ContactEventListener.contactModified (int event, int contactIdentifier)* each time a contact is modified.
- *ContactEventListener.groupModified (int event, int groupIdentifier, int contactIdentifier)* each time a group is modified.
- ContactEventListener.fieldDescriptorModified (int event, int fieldDescriptorIdentifier) each time a field descriptor is modified.

4.3.2 Definition of events

The following events can notify an applet:

Table 1: Contact Manager events list

Event Name	
EVENT_CONTACT_ADDED	
EVENT_CONTACT_MODIFIED	
EVENT_CONTACT_REMOVED	
EVENT_FIELD_DESCRIPTOR_ADDED	
EVENT_FIELD_DESCRIPTOR_MODIFIED	
EVENT_FIELD_DESCRIPTOR_REMOVED	
EVENT_GROUP_ADDED	
EVENT_GROUP_CONTACT_ADDED	
EVENT_GROUP_CONTACT_REMOVED	
EVENT_GROUP_REMOVED	

4.4 Services invocation

4.4.1 General

Before contacts can be created and used, the Contact Manager Server shall first be configured with the contacts structure (i.e. the field descriptors).

The Contact Manager internal interface provides services to access and manage contacts, groups of contacts, list of contacts, list of groups, and contacts structure.

In order to access the Contact Manager Server, an application shall use the factory methods in the *ContactManager* class. This class, which is the primary entry point to the API, provides the applications with factory methods to create all objects needed to access the Contact Manager Server.

Note: in its current form, it allows only one instance of the contact manager data model. However, all interfaces in this API are designed such that multiple contact manager information bases could co-exist.

With the exception of the *ContactManager* factory class and the *ContactManagerException* class, all objects defined in the data model are accessed using interfaces.

4.4.2 Services invocation API description

The invocation of the Contact Manager Server services is done through methods corresponding to the Contact Manager Server Runtime Environment interface.

4.4.2.1 View interfaces

All objects in the Contact Manager Server"s data model are accessed using the View interfaces instances:

- The *FieldDescriptorsListView* interface shall be used by client applications to access the field descriptors list.
- The *FieldDescriptorView* interface shall be used by client applications to access the field descriptors.
- The *ContactsListView* interface shall be used by client applications to access the contacts list.
- The *GroupsListView* interface shall be used by client applications to access the groups list.
- The *ContactView* interface shall be used by client applications to access the contacts.
- The *FieldView* interface shall be used by client applications to access the fields.

Client applications should obtain instances of these interfaces at installation time.

The instances of *FieldDescriptorsListView*, *ContactsListView* and *GroupsListView* each gives access to a single collection during their whole lifetime, respectively the unique list of all field descriptors, the unique list of all contacts, and the unique list of all groups.

Conversely, each instance of *FieldDescriptorView*, *ContactView* and *FieldView* may, at different times, give successive access to distinct objects of the Contact Manager Server"s database. At some times, they may give access to none.

At any time, each of these latter views instances are either in DESELECTED or in SELECTED state:

- A view in DESELECTED state does not represent any existing object in the Contact Manager Server"s data model.
- A view in SELECTED state represents an existing object in the Contact Manager Server"s data model. It can be used to access this selected object.

4.4.2.2 Lists access interfaces

The interfaces *ContactsListView*, *GroupsListView*, *FieldDescriptorsListView*, *ContactView* and *GroupView* provide access to collections:

- *ContactsListView* provides access to a collection of contacts;
- *GroupsListView* provides access to a collection of Groups;
- *FieldDescriptorsListView* provides access to a collection of field descriptors;
- *ContactView* provides access to a collection of fields;
- *GroupView* provides access to a collection of contacts.

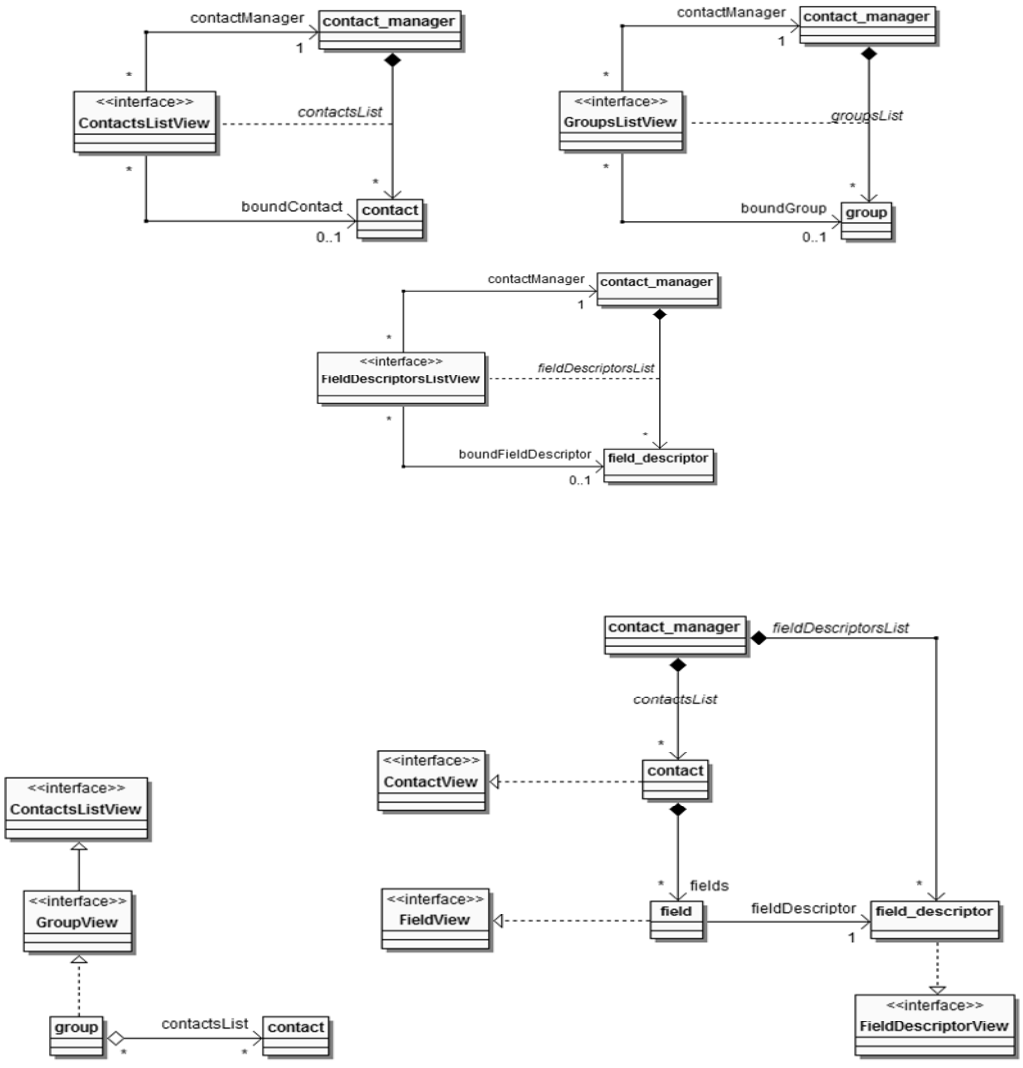

Figure 1: interfaces

These interfaces provide methods to enumerate items in these collections, as well as methods to add and remove items.

4.4.2.3 Items update interfaces

The groups, contacts, fields and field descriptors can be accessed using the interfaces *GroupView*, *ContactView*, *FieldView* and *FieldDescriptorView*, respectively.

These interfaces shall be used to update the objects they represent.

The life cycle of the *ContactView*, *FieldView*, *GroupView* and *FieldDescriptorView* is defined by the following state diagram:

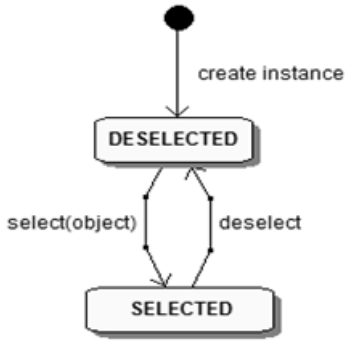

Figure 1: state diagram

- In the DESELECTED state, the instance does not represent any existing group, contact, field or field descriptor in the contact manager. It can be used to store values for a new object to be created subsequently.
- In the SELECTED state, the instance represents an group, contact, field or field descriptor in the contact manager.

The select operation occurs when the following methods are invoked:

Interface	select occurs when the following methods are invoked
Group*View*	*GroupsListView.selectGroup()*
	GroupsListView.selectNextGroup()
	GroupsListView.addGroup()
Contact*View*	*ContactsListView.selectContact()*
	ContactsListView.selectNextContact()
	ContactsListView.addContact()
	GroupView.selectContact()
	GroupView.selectNextContact()
Field*View*	*ContactView.selectField()*
	ContactView.selectNextField()
FieldDescriptor*View*	*FieldDescriptorsListView.selectFieldDescriptor()*
	FieldDescriptorsListView.selectNextFieldDescriptor()
	FieldDescriptorsListView.addFieldDescriptor()

The deselect operation occurs when the following events occur:

- UICC RESET
- *deselect()* method of the appropriate View is invoked.

4.4.2.4 Access control

The API shall only accept requests from trusted applets, defined as follows:

- Security Domains of UICC applications defined in chapter 4.3 of [2] are trusted by the API.
- Any applet of any trusted Security Domain is trusted by the API.

4.4.3 API classes and interfaces detailed description

Annex A provides the detailed description of the classes and interfaces that constitute the Services Invocation API.

Annex A (normative): Java Card™ platform Contact Manager API

The attached files "31221_Annex_A_Java.zip", and "31221_Annex_A_HTML.zip" contains source files and html documentation for the Java Card™ Contact Manager API.

Annex B (normative): Java Card™ platform Contact Manager API identifiers

The attached file "31221_Annex_B_Export_files.zip" contains the export files for the uicc.contactmanager.* package.

Annex C (normative): Java Card™ platform Contact Manager API package version management

The following table describes the relationship between each TS 31.221 specification version and its packages AID and Major, Minor versions defined in the export files.

Table 1

TS 31.221	uicc. contactmanager package	
	AID	Major, Minor
	A0 00 00 00 87 10 07 FF FF FF FF 89 14 10 00 00	2.0

The package AID coding is defined in ETSI TS 101 220 [7]. The Contact Manager 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): Example of Java Card™ platform Contact Manager API use

The attached file "31221_Annex_D_Example.zip" contains an example of use of the Java™ Card Contact Manager API.

Annex E (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
2008-12	CT #42	CP-080900			Approval of the specification	2.0.0	8.0.0
2009-01	---------				Re-publication with attached Java™ annexes	8.0.0	8.0.1
2009-03	CT #43	CP-090195	0001		Correction of ContactEventListener interface	8.0.1	8.1.0
2009-12	CT #46				Upgrade to Rel-9 with no changes (MCC)	8.1.0	9.0.0
2010-12	CT #50	CP-100836	0003	1	Update reference to "Java Card 3.0.1 Classic" reference	9.0.0	9.1.0
2011-04	CT #51				Upgrade to Rel-10 with no changes (MCC)	9.0.0	10.0.0
2012-09	CT #52				Upgrade to Rel-11 with no changes (MCC)	10.0.0	11.0.0

History

Document history		
V11.0.0	November 2012	Publication