ETSI TS 102 267 V18.0.0 (2023-01)

Smart Cards;
Connection Oriented Service API for the Java Card™ platform
(Release 18)

Reference
RTS/SET-T102267vi00

Keywords
API, protocol, smart card, testing, transport

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 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are 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 Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, 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.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™** and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Secure Element Technologies (SET).

The contents of the present document are subject to continuing work within TC SET and may change following formal TC SET approval. If TC SET 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 SET for information;
  - 2 presented to TC SET for approval;
  - 3 or greater indicates TC SET 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 an Application Programming Interface for the Java Card™ to use transport protocols (e.g. CAT_TP as defined in ETSI TS 102 127 [1]) for CAT applications.

This stage 2 document describes the interface functionalities, the interface working mechanisms and its information flow.

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 https://docbox.etsi.org/Reference/.

- In the case of a reference to a TC SET 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] ETSI TS 102 127: "Smart Cards; Transport protocol for CAT applications; Stage 2".

[2] ORACLE: "Java Card™ Platform, Virtual Machine Specification, Classic Edition, Version 3.1".

NOTE: Available at http://docs.oracle.com/en/java/javacard/3.1/.

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

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

[5] GlobalPlatform™: "GlobalPlatform Technology, Remote Application Management over HTTP, Card Specification v2.3- Amendment B", Version 1.2.

NOTE: Available at http://www.globalplatform.org/.

[6] ETSI TS 102 483: "Smart cards; UICC-Terminal interface; Internet Protocol connectivity between UICC and terminal".

[7] ETSI TS 102 225: "Smart Cards; Secured packet structure for UICC based 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 SET 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:

BIPLink: interface to access the physical layer by means of the Bearer Independent Protocol according to ETSI TS 102 223 [4]

transport layer: instance within the card framework which implements the transport protocol, e.g. CAT_TP

3.2 Symbols

Void.

3.3 Abbreviations

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

ACK	ACKnowledge
AID	Application IDentifier
APDU	Application Protocol Data Unit
API	Application Programming Interface
BIP	Bearer Independent Protocol

NOTE: See ETSI TS 102 223 [4].

CAT	Card Application Toolkit

NOTE: See ETSI TS 102 223 [4].

CAT_TP	Card Application Toolkit Transport Protocol

NOTE: See ETSI TS 102 127 [1].

HTTP	HyperText Transfer Protocol
IP	Internet Protocol
JCRE	Java Card™ Runtime Environment
PDU	Packet Data Unit
RAM	Remote Application Management
RFM	Remote File Management

SCP81 Secure Channel Protocol '81'

NOTE: See "GlobalPlatform Technology, Remote Application Management over HTTP, Card Specification v2.3 Amendment B" Version 1.2 [5].

SCWS Smart Card Web Server
SDU Service Data Unit
SMS Short Message Service
TCP Transmission Control Protocol
TLS Transport Layer Security

4 Overall description

4.0 General

The present document describes an API that provides applications a set of Connection Oriented Services. This API provides either direct access to the transport protocols supported by the terminal (by using the BIP) or access to transport protocol layers provided by the UICC, e.g. the CAT_TP protocol layer in the UICC or SCP81.

4.1 Connection API concept

The API is based on the concept of objects that encapsulate the features of connection service (e.g. opening a service, sending and receiving data, etc.).

The present document provides three variants of connection services to the application. These variants are based on the *Connection* interface which is an abstraction of the BIP channel which is used to exchange data.

One type of connection service is accessible through the *BIPLink* interface. It provides direct access to the transport protocol stack in the terminal. Another type of connection is accessible via the *UICCTransportLink* interface. It provides an interface to a transport protocol layer deployed on the UICC, e.g. CAT_TP. The transport protocol layer on the UICC uses BIP to transport its PDU to the terminal. The last type of connection is accessible through the *SCP81Connection* interface. It provides an interface to a transport protocol layer based on SCP81 as defined in [5].

Implementations of *BIPLink* interface and *UICCTransportLink* interface are based on the *Observer* design pattern. I.e. the application is notified about state changes (e.g. data received, channel closed, etc.) by means of events sent to the *Observer* interface.

The support of each type of connection is optional. If a type of connection is not supported, the related method in the *ConnectionServer* class shall throw a *ConnectionException* with reason code *TRANSPORT_PROTOCOL_NOT_SUPPORTED.*

5 API architecture

5.0 General

For a reference documentation of all classes and interfaces of the Connection oriented Services API refer to annex A.

5.1 API usage

5.1.1 Establishing a Connection

To perform operations on a specific link it is required to establish a *Connection;* a *Connection* can be considered as a pure point to point connection on which a link can be established.

Depending on the nature of the link, the *Connection* may be shared among different links or can be used exclusively by one link. In case of sharing, all the *Connection* shall have the same parameters. Even if the links allow to share the same *Connection*, it is possible, at *Connection* instantiation time, to specify that a *Connection* cannot be shared among several links.

The *Connection* may require some specific Toolkit resources to be available to the application, e.g. in case of BIP connections, at least one BIP channel shall be available to the application when the *Connection* is opened.

5.1.2 Opening a BIPLink

A *BIPLink* requires the usage of an underlying *Connection* to transport data according to the BIP protocol. The used *Connection* shall be a BIP connection.

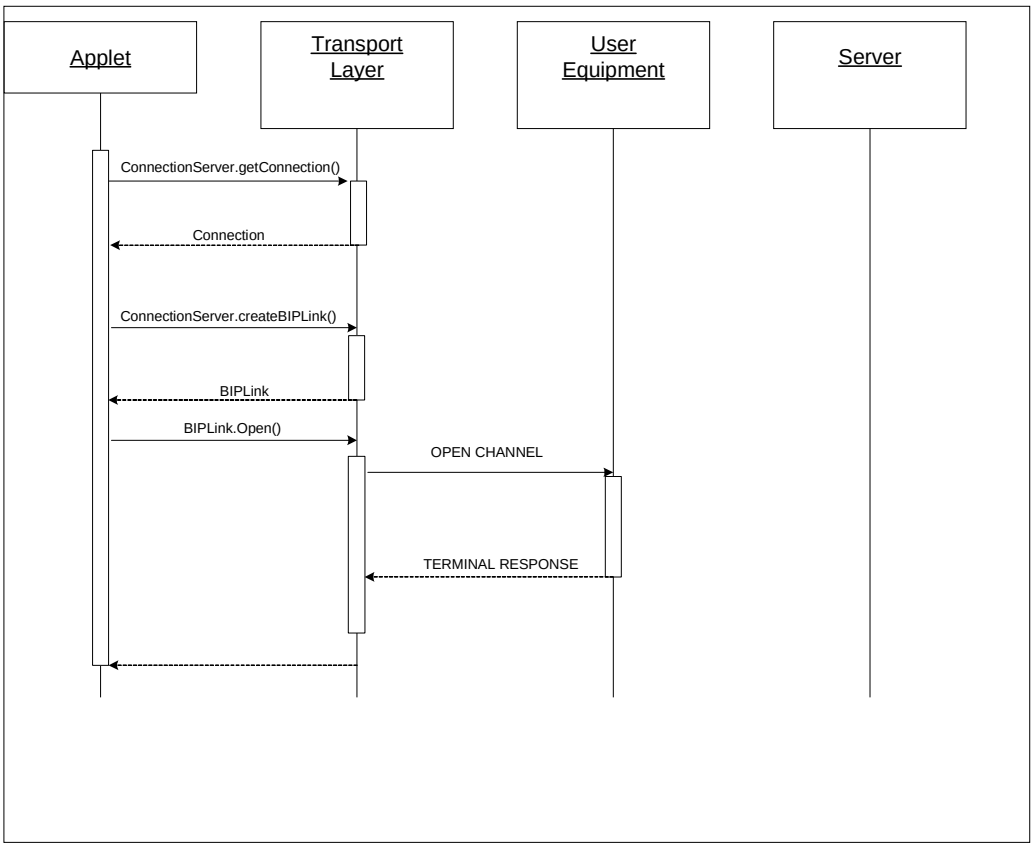

Figure 1: Opening a BIPLink

5.1.3 Creating an UICC Transport link

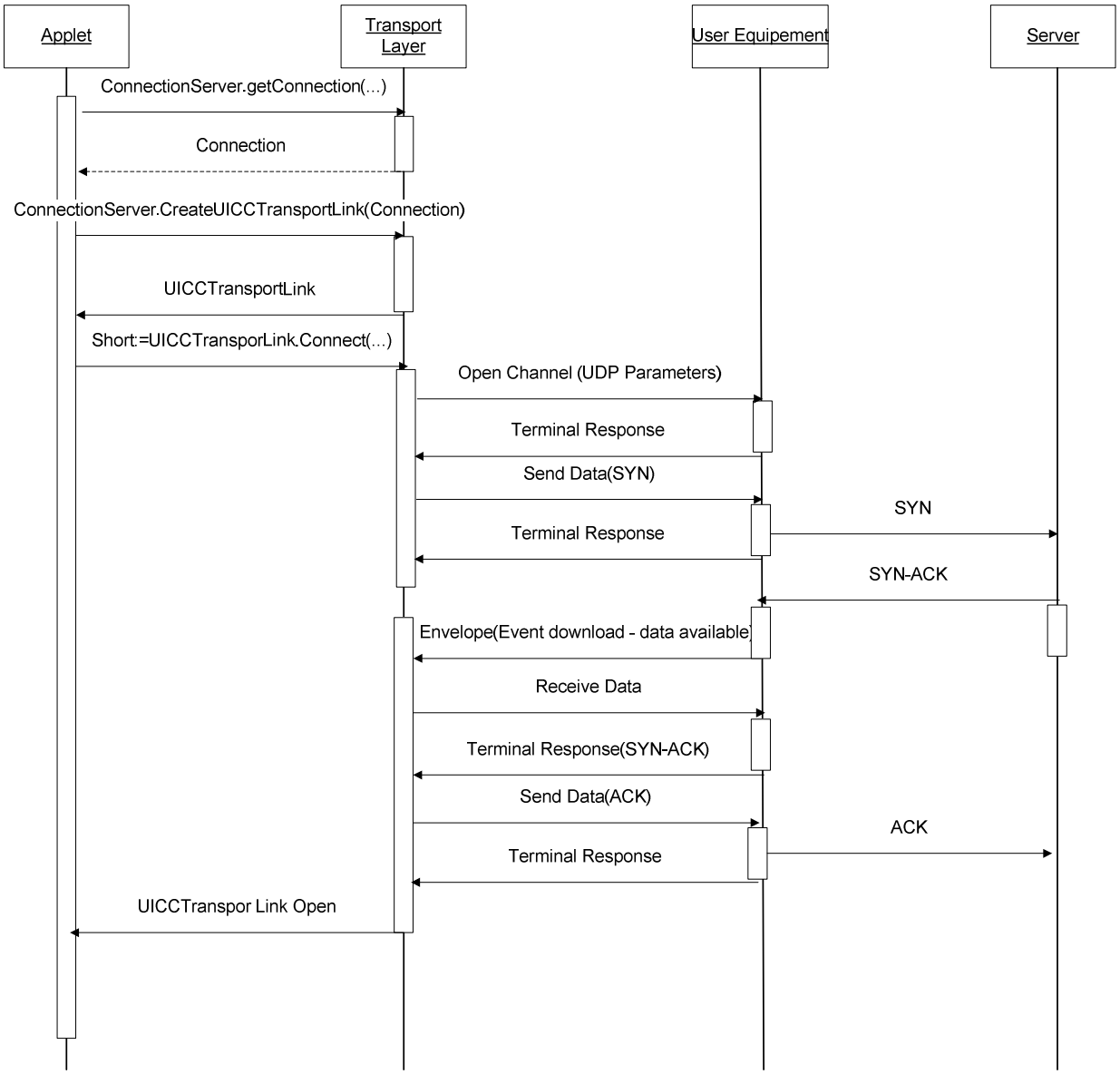

Figure 2: Creating a reliable link

The API is considered as a blocking API which means that no other applet is able to use the *uicc.toolkit.ProactiveHandler* while the call to the method *UICCTransportLink.connect()* is ongoing, i.e. the three way handshake is performed.

5.1.4 Sending data over UICC Transport Link

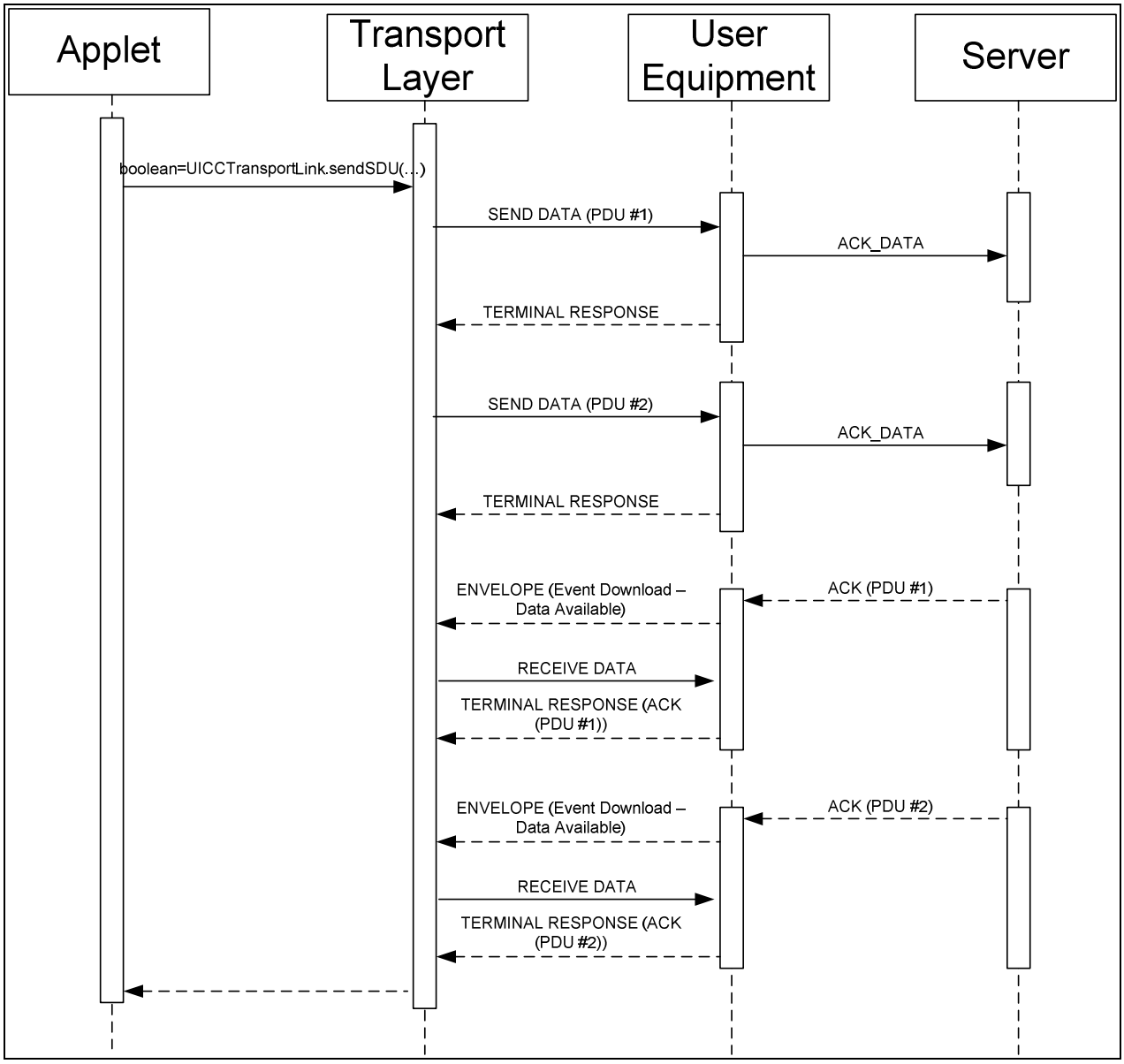

Figure 3: Sending data over Reliable Link

No other applet is able to use the *uicc.toolkit.ProactiveHandler* while the call to the method *UICCTransportLink.sendSDU*(..) is ongoing, i.e. that each PDU has been acknowledged.

After UICCTransportLink.sendSDU method returns, the uicc.toolkit.ProactiveHandler shall be available to the calling applet if it was available before the method invocation; however, handler content may be affected by the system during method execution.

5.1.5 Receiving data

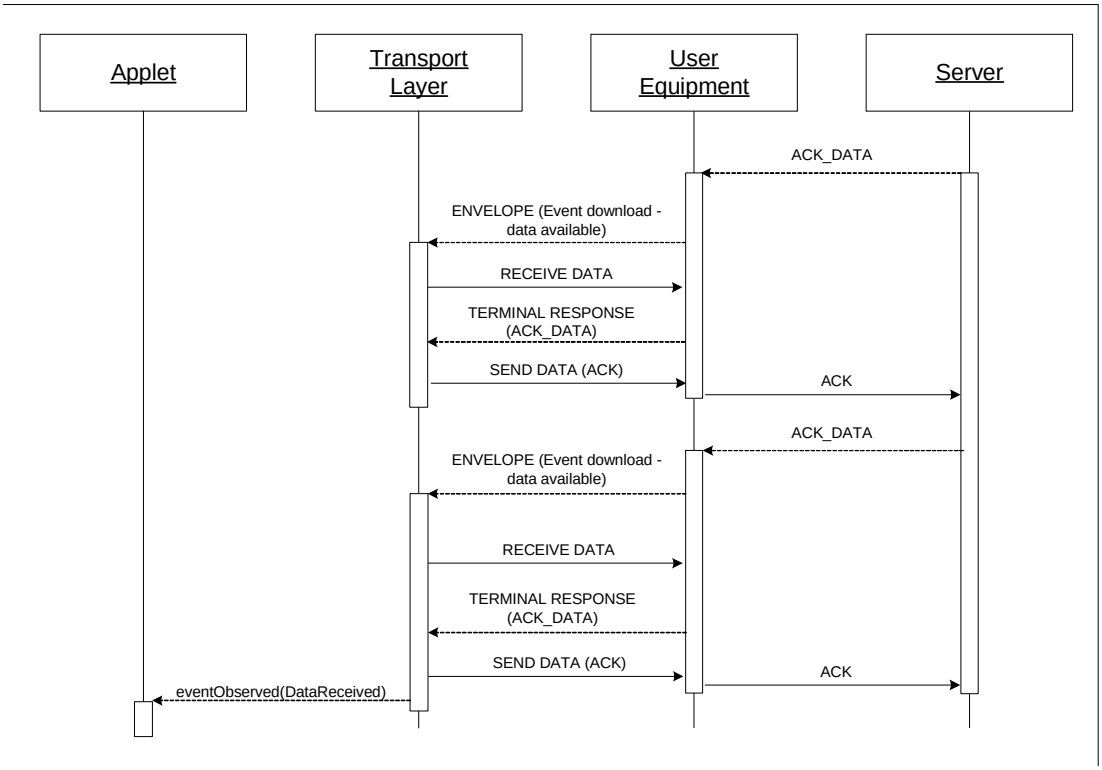

Figure 4: Receiving data

Upon reception of an ENVELOPE (Event Download - Data available), the *uicc.toolkit.ProactiveHandler* is locked by the Transport Layer to perform RECEIVE DATA proactive commands; once the terminal indicates that all the available data have been received, the *uicc.toolkit.ProactiveHandler* is released by the Transport Layer to let other applications to perform proactive commands.

This behaviour might be affected by other applications, as e.g. a SMS data download which causes a proactive session might result in a CAT_TP time-out.

5.2 Multiplexing through one BIP connection

If a *Connection* object has been created with the *multiplexingAllowed* property, it can be used by several reliable links.

The *Connection* status is shared by all *ReliableLink* objects using it.

Multiplexing is not allowed for *BIPLink* objects.

5.3 Behaviour in duplex communication

5.3.1 Receiving data while sending data

Expected behaviour when terminal sends Envelope (Event Download- data available), either on the same BIP channel where the sending of data is ongoing or for another BIP channel:

- The reliable link layer shall receive data and handle the transport protocol (e.g. send ACK). The reliable link layer shall not trigger any application on reception of a SDU before the SDU has not been acknowledged.
- The reliable link protocol implementation (e.g. CAT_TP protocol implementation) handles buffer space by adjusting window size for active CAT_TP connections.

5.3.2 Sending data while reception of data is ongoing

In line with the behaviour defined in the previous clause sending while receiving is possible and shall be supported.

5.3.3 Receiving data of more than 1 connection simultaneously

This shall be supported by the CAT_TP API implementation.

5.3.4 Sending data before applications returns from DataReceived

This case shall be allowed. Note that no conflict can occur because the application will not be triggered while a call to the blocking ReliableLink.send() method is ongoing.

5.4 Interference with other proactive commands

Proactive command (like e.g. Select Item) sent from any application may block the toolkit protocol and thus prevent the card from receiving PDUs. This may result in timeouts in the CAT_TP protocol. This behaviour is caused by limitations of the Toolkit protocol and cannot be avoided.

While sending data the CAT_TP layer shall lock the Proactive Handler to avoid that other applications interfere with the sending of CAT_TP data. This behaviour is required in order to be able to receive the ACK PDU from the server in order to avoid retransmissions.

5.5 Secured communication based on SCP81

5.5.0 General

It is possible for an application to open a connection based on SCP81 to a remote server. The security for data exchange is provided by TLS. The HTTP protocol is used on top of TLS to provide encapsulation of the data.

TCP/IP transport is provided by the Bearer Independent Protocol of ETSI TS 102 223 [4] or a direct IP connection as specified in ETSI TS 102 483 [6].

The processing rules for messages that are protected using SCP81 are specified in Amendment B of the Global Platform Card Specification v2.3 [5] and in ETSI TS 102 225 [7]. Application data shall be sent in the body part of HTTP POST requests and HTTP POST responses. The specialized processing rules for transport of remote APDUs as detailed in Amendment B of the Global Platform Card Specification v 2.3 [5], clauses 3.3.3, 3.4, 3.5, 3.6, 3.7 and Annex A do NOT apply. The following rules apply:

- The Content-Type header field of the HTTP POST request is set by the card application. It shall start with:
  - "application/vnd.etsi.scp.request." or "application/vnd.etsi.scp.request;".
- The Content-Type header field of the HTTP POST response is set by the remote server. It shall start with:
  - "application/vnd.etsi.scp.response." or "application/vnd.etsi.scp.response;".
- The X-Admin-Script-Status and X-Admin-Targeted-Application headers are meaningless and shall be missing in all HTTP POST responses/requests of the protocol defined in this clause.
- The first request of a session may contain a body if requested by the card application.

The framework shall reject requests from applications that includes a Content-Type header field not conforming to the rule above.

The framework shall ignore additional header fields (including X-Admin-Script-Status and X-Admin-Targeted-Application) in messages from the remote server for the application.

The framework shall process messages from the remote server that include a Content-Type header field not conforming to the rule above as follows:

- If the Content-Type set by the remote server refers to another protocol (see below), the application is notified of such an event by means of the *SCP81ConnectionStatus* before switching to the other protocol.

- Otherwise the application shall be invoked with the incoming data. The application can detect if the Content-Type header field was correct via the method *contentTypeIsCorrect()*.

The communication is handled by the entity which implements the *SCP81Connection* interface.

TLS Security is handled by the associated (directly or indirectly) Security Domain of the card application which has initiated the SCP81Connection.

When all headers of an HTTP response have been received by the card, the registered *Observer* is notified with the *DataReceived* event irrespective of the presence of a body in the HTTP response. The application can read the HTTP response with invocation of the non-blocking method *DataReceived.copyReceivedData()*. The registered *Observer* is also notified when new data has been received belonging to the same HTTP POST response (e.g. transfer coding chunked is used).

It is possible to switch from the protocol described above to RAM/RFM over HTTP or to SCWS remote administration protocol based on the Content Type set by the remote server as specified in ETSI TS 102 225 [7].

NOTE: This switch works only if the applications which handle RAM/RFM over HTTP or SCWS remote administration protocol are directly or indirectly associated with the same Security Domain which handles the SCP81 secure channel.

It is not possible to switch from RAM/RFM over HTTP or SCWS remote administration protocol to the one described above.

5.5.1 Sequence diagram

Figure 5 presents a sequence diagram describing the *SCP81Connection* interface usage.

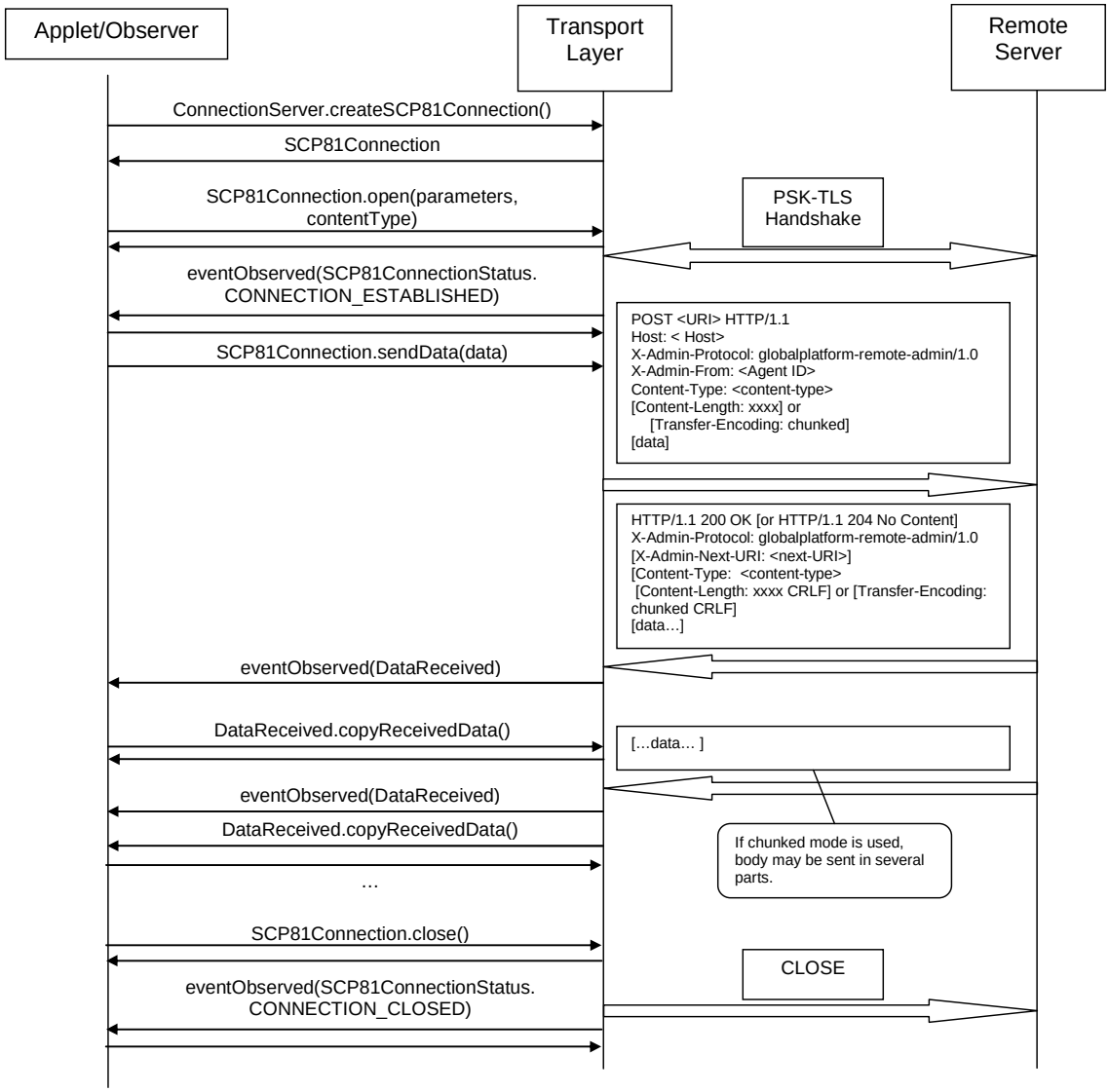

Figure 5: SCP81Connection sequence diagram

The remote server behaviour is out of the scope of the present document and is application dependant.

Annex A (normative): Connection API

The source files for the uicc.connection package (102267_Annex_A_Java.zip and 102267_Annex_A_HTML.zip) are contained in ts_102267v180000p0.zip which accompanies the present document.

Annex B (normative): Connection API identifiers

The export files for the uicc.connection package (102267_Annex_B_Export_Files.zip) are contained in ts_102267v180000p0.zip which accompanies the present document.

NOTE: See the "Java Card Platform, Virtual Machine Specification, Classic Edition, Version 3.1" [2]. It should be noted that the CAP and Export file format version, defined in the "Java Card Platform, Virtual Machine Specification, Classic Edition, Version 3.1" [2] was updated to the version 2.3. Implementations that support these new versions are backward compatible with earlier versions of the Java Card Platform specification.

Annex C (normative): Connection API package version management

Table C.1 describes the relationship between each ETSI TS 102 267 specification version and its Connection Oriented Service API packages AID and Major, Minor versions defined in the export files.

Table C.1

uicc.connection package		
ETSI TS 102 267	**Major, Minor**	**AID**
7.0.0	1.0	A000000009 0005 FFFF FFFF 89 15 000000
11.0.0	2.0	

The package AID coding is defined in ETSI TS 101 220 [3]. The Connection Oriented Service API package AID is 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.

For a table describing the versioning of a package, a line is introduced only upon changes of Major or Minor version of its package.

Annex D (informative): Bibliography

- ORACLE: "Java Card Platform, Java Card API, Classic Edition, Version 3.1".
- ORACLE: "Java Card Platform, Runtime Environment Specification, Classic Edition, Version 3.1".
- ETSI TS 102 221: "Smart Cards; UICC-Terminal interface; Physical and logical characteristics".
- ETSI TS 102 241: "Smart Cards; UICC Application Programming Interface (UICC API) for Java Card™".

Annex E (informative): Change history

This annex lists all changes made to the present document.

Change History								
Date	**Meeting**	**Tdoc**	**CR**	**Rv**	**Cat**	**Changes**	**Old**	**New**
2010-03	SCP#43	SCP-090327	001	-	F	Correction of the AID of the Connectivity API. Correction of erroneous clause numbering	7.0.0	7.1.0
2011-12	SCP-53	SCP(11)0378	002	-	B	Definition of a new API to allow applications to use Secure messaging over HTTP(S)	10.0.0	11.0.0
2012-12	SCP-57	SCP(12)000263	003	-	F	Correction of HTTP header fields	11.0.0	11.1.0
2014-09	SCP-65	SCP(14)000211	003	-	D	Update of Java Card reference	11.1.0	12.0.0
2016-04	SCP-73	SCP(16)000090r1	004	1	F	Update of references to GlobalPlatform specifications	12.0.0.	13.0.0
2019-01						Update to Release 14	13.0.0	14.0.0
2018-04	SCP-83	SCP(18)000086	005		D	Annex C table reformat	14.0.0	15.0.0
2021-03						Automatic Upgrade	15.0.0	16.0.0
2020-12	SCP-97	SCP(20)000158	006		C	Update the reference of Java Card™ specifications to the latest release	15.0.0	17.0.0
2022-07	SET-106	SET(22)000099r1	007	1	B	Update of references to GlobalPlatform specifications	17.0.0	18.0.0

History

Document history		
V18.0.0	January 2023	Publication