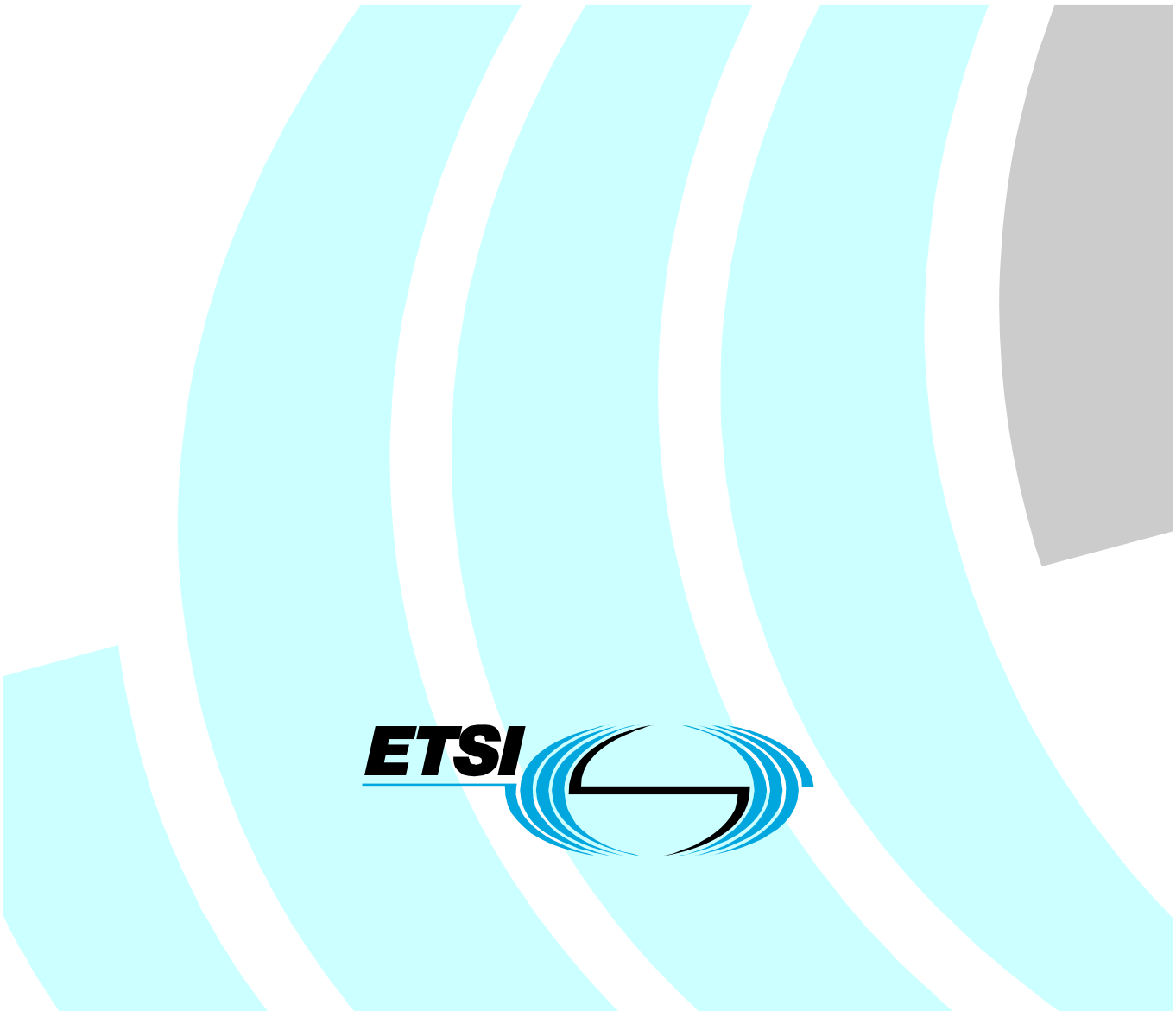
ETSI TS 102 267 V7.0.0 (2007-08)

Technical Specification

Smart Cards;
Connection Oriented Service API for the Java Card™ platform (Release 7)



Reference
DTS/SCP-T006a

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 - 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/chaircor/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 2007.
All rights reserved.

DECT™, **PLUGTESTS**™ and **UMTS**™ are Trade Marks of ETSI registered for the benefit of its Members.
TIPHON™ and the **TIPHON logo** are Trade Marks currently being registered by ETSI for the benefit of its Members.
3GPP™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners.

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

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

The 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 defines an Application Programming Interface for the Java Card™ to use transport protocols (e.g. CAT_TP as defined in 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

References are either specific (identified by date of publication and/or edition number or version number) or non-specific.

- For a specific reference, subsequent revisions do not apply.
- In the case of a reference to an TC SCP document, a non specific reference implicitly refers to the latest version of that document in the same Release as the present document.
- Non-specific reference may be made only to a complete document or a part thereof and only in the following cases:
  - if it is accepted that it will be possible to use all future changes of the referenced document for the purposes of the referring document;
  - for informative references, the latest version applies. In the case of a reference to an 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.

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

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

2.1 Normative references

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

[1] ETSI TS 102 127: "Smart Cards; Transport protocol for CAT applications; Stage 2 (Release 6)".

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

NOTE: SUN Java Card Specifications can be downloaded at http://java.sun.com/products/javacard.

[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)".

2.2 Informative references

[5] Sun Microsystems Java CardTM Specification: "Java CardTM 2.2.2 Application Programming Interface".

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

[7] ETSI TS 102 221: "Smart Cards; UICC-Terminal interface; Physical and logical characteristics".

[8] ETSI TS 102 241: "Smart Cards; UICC Application Programming Interface (UICC API) for Java Card (TM)".

3 Definitions and abbreviations

3.1 Definitions

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

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

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

3.3 Abbreviations

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

AID Application IDentifier
API Application Programming Interface
BIP Bearer Independent Protocol
NOTE: See TS 102 223 [4].

CAT Card Application Toolkit
NOTE: See TS 102 223 [4].

CAT_TP Card Application Toolkit Transport Protocol
NOTE: See TS 102 127[1].

FFS For Further Study
JCRE Java Card™ Runtime Environment

4 Overall description

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.

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, …).

The present document provides two variants of connection services to the application. Both 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. The other 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.

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.

5 API architecture

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 must be available to the application when the *Connection* is opened.

5.1.1 Opening a BIPLink

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

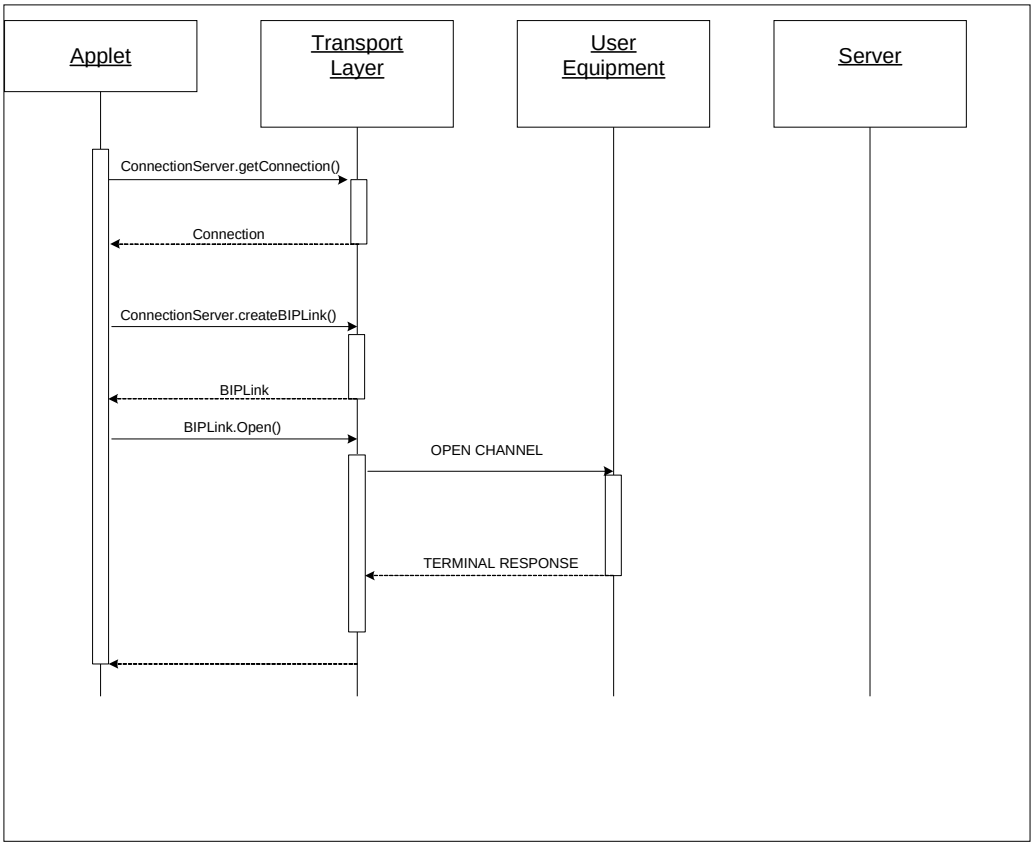


Figure 1: Opening a BIPLink

5.1.2 Creating an UICC Transport link

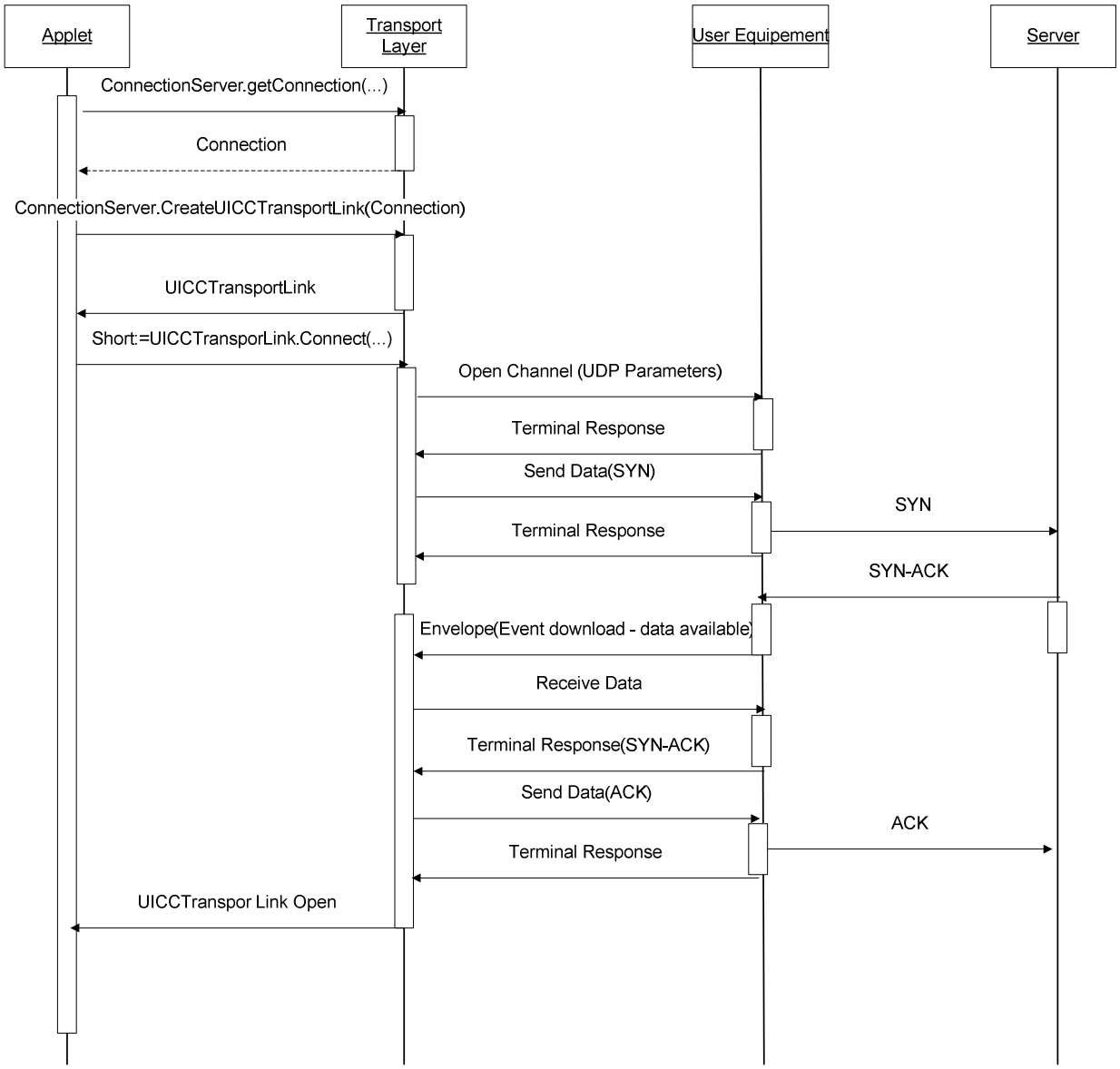


Figure 2: Creating a reliable link

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.

5.1.3 Sending data over UICC Transport Link

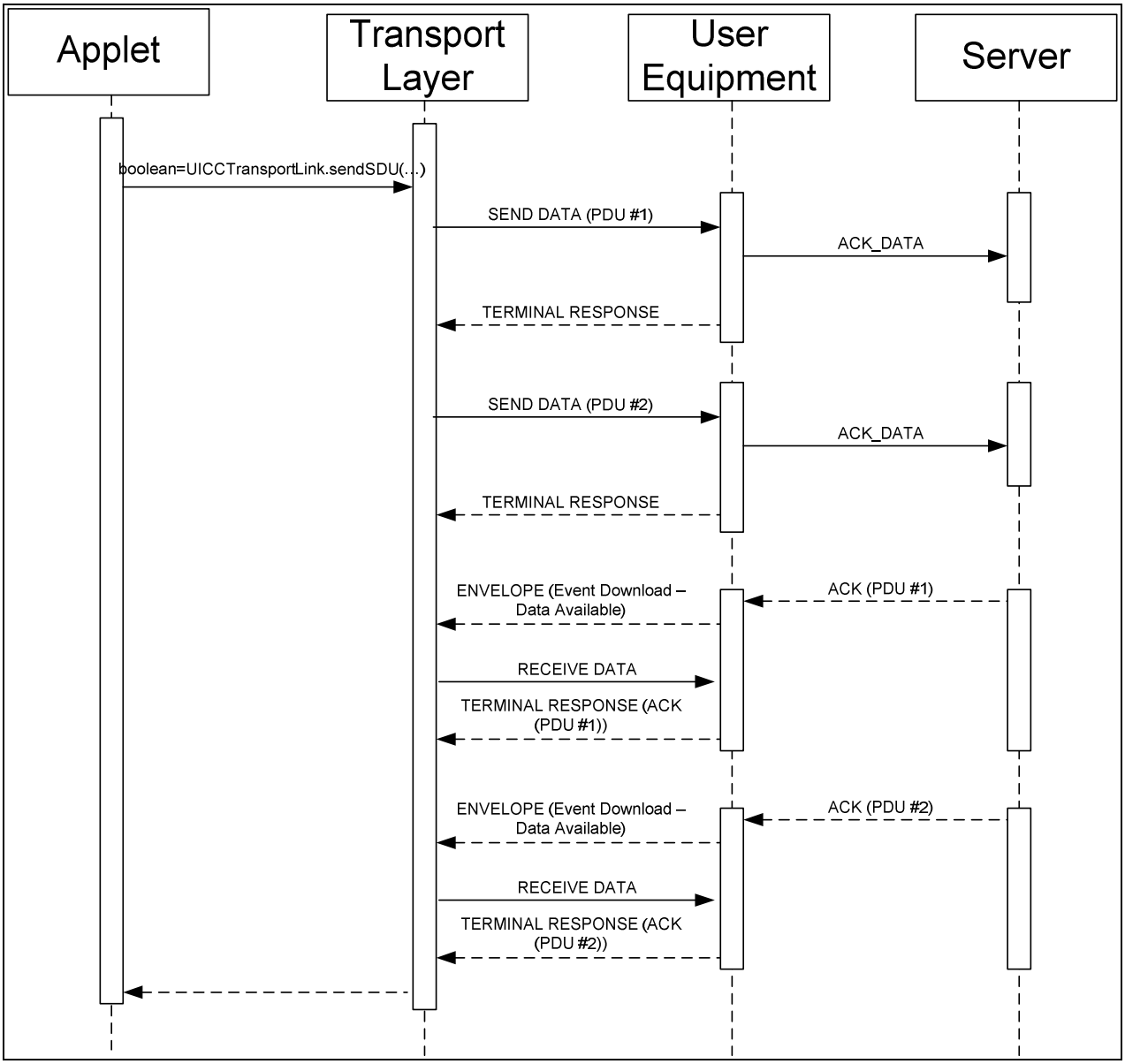


Figure 3: Sending data over Reliable Link

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.

5.1.4 Receiving data

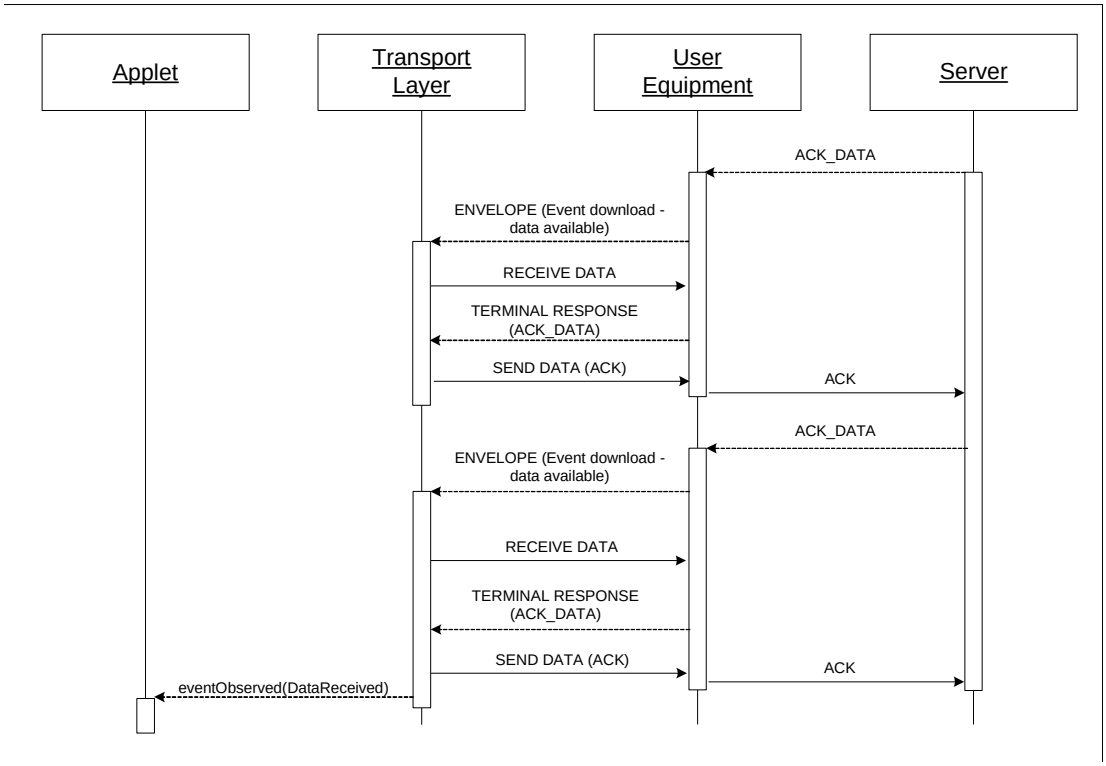


Figure 4: Receiving data

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.

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_102267v070000p0.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_102267v070000p0.zip which accompanies the present document.

NOTE: See the "Java Card™ 2.2.2 Virtual Machine Specification" [2].

Annex C (normative): Connection API package version management

Table C.1 describes the relationship between each 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

TS 102 267	uicc.connection package	
	AID	Major, Minor
	A0 00 00 00 09 00 05 FF FF FF FF 89 14 00 00 00	1.0

The package AID coding is defined in 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.

History

Document history		
V7.0.0	August 2007	Publication