TRUSTED CONNECTIVITY ALLIANCE

Implementation Guideline for the UICC – Terminal USB Interface

Version 0.9

Published by simalliance now Trusted Connectivity Alliance

October 2008

Contents

Figure index

1 Introduction

After decades of use in all relevant smartcard applications a successor to the asynchronous transmission protocol defined by ISO 7816-3 [13] has been selected by the European Telecommunications Standardisation Institute (ETSI) which is now being introduced into the market for mobile communications: the USB Inter-Chip interface.

The ISO 7816-3 interface was well suited for smart cards with limited memory and processing resources and for applications which integrated basic smart card cryptographic functions. The evolving technology in particular in the area of semiconductors and communication and an increasing market demand for more complex applications requiring larger amounts of data to be transferred made the introduction of a new interface inevitable. With the introduction of the new USB interface smart cards will be capable to offer security and data services over state-of-the-art internet protocols thus seamlessly integrating into today's IT infrastructure.

The change to a central component of a standardized system always bears interoperability risks. The main purpose of this document therefore is to reduce this risk and to contribute to a faster and smoother introduction of USB UICCs and USB terminals into the market. The main chapters of this document are dedicated to this task and give guidance on the implementation of UICC hardware and the three USB communications classes relevant to the new generation of USB cards and terminals: the ICCD or APDU Class, the Mass Storage Class and the CDC EEM (or IP connectivity) Class.

Further and to the benefit of those readers who have not been able to follow the discussion about the use cases behind the introduction of the new interface an Annex of this document gives background information showing the relevance of the technology to the most important applications like local content browsing (using the Smart Card Web Server), handset customization and contact management.

2 Reference Documentation

[1] 3GPP TS 31.111 "USIM Application Toolkit (USAT)" (Release 7)

[2] 3GPP TS 31.102 "Characteristics of the USIM application" (Rel. 7)

[3] ETSI TS 102 221 "Smartcards, UICC-Terminal Interface; Physical and logical Characteristics" (Rel. 7)"

[4] ETSI TS 102 223 "Smartcards, Card Application Toolkit (CAT)" (Rel. 7)

[5] ETSI TS 102 483 "Smart Cards;UICC-Terminal interface; Internet Protocol connectivity between UICC and Terminal" (Rel. 7)

[6] ETSI TS 102 600 "Smart Cards; UICC-Terminal interface; Characteristics of the USB interface" (Rel. 7)"

[7] ETSI TS 102 613 "UICC-CLF interface; Physical and logical characteristics" (Rel. 7)

[8] ETSI TS 102 622 "UICC-CLF interface Host Controller Interface" (Rel. 7)

[9] INCITS 408-2005, "SCSI Primary Commands - 3 (SPC-3)", available at http://www.t10.org.

[10] INCITS 405-2005, "SCSI Block Commands - 2 (SBC-2)", available at http://www.t10.org.

[11] ISO/IEC 7816-1: "Identification cards - Integrated circuit(s) cards with contacts - Part 1: Physical characteristics".

[12] ISO/IEC 7816-2:1999/ AM1: 2004, "Identification cards - Integrated circuit(s) cards with contacts - Part 2: Dimension and location of the contacts, Amendment 1: Assignment of contacts for C4 and C8".

[13] ISO/IEC 7816-3: "Information technology - Identification cards - Integrated circuit(s) cards with contacts - Part 3: Electronic signals and transmission protocols".

[14] ISO/IEC 7816-12: "Identification cards - Integrated circuit cards - Part 12: Card with contacts – USB electrical interface and operating procedures".

[15] ISO/IEC 9293, "Volume and file structure of disk cartridges for information interchange".

[16] Universal Serial Bus, Device Class Specification for Mass Storage Devices, "Mass Storage Bulk Only specification", Revision 1.00, USB Implementers Forum, Device Working Group: Mass Storage.

Available at http://www.usb.org/developers/devclass_docs

[17] Universal Serial Bus, "Mass Storage Class Specification Overview", Revision 1.2, USB Implementers Forum, Device Working Group: Mass Storage.

Available at http://www.usb.org/developers/devclass_docs

[18] Universal Serial Bus, Device Class Specification for Mass Storage Devices, "Mass Storage Bootability specification", Revision 1.0, USB Implementers Forum, Device Working Group: Mass Storage, Available at http://www.usb.org/developers/devclass_docs

[19] Universal Serial Bus, "Device Class Specification for USB Integrated Circuit Card Devices" (Smart Card ICCD), Revision 1.00, USB Implementers Forum, Device Working Group: Smart Cards.

Available at http://www.usb.org/developers/devclass_docs

[20] Universal Serial Bus, Device Class Specification for Communication Devices, "CDC Subclass specification for Ethernet Emulation Model" subclass, Revision 1.00, USB Implementers Forum, Device Working Group: Communication.

Available at http://www.usb.org/developers/devclass_docs

[21] Universal Serial Bus Specification Revision 2.0, USB Implementers Forum.

Available at http://www.usb.org/developers/docs.

This is a ZIP file package containing the following:

- The original USB 2.0 Core specification released on April 27, 2000
- The USB On-The-Go supplement Revision 1.2 as of April 4, 2006
- The Inter-Chip USB supplement Revision 1.0 as of March 13, 2006
- Errata and Engineering Change Notices

[22] Interface Association Descriptors, USB ENGINEERING CHANGE NOTICE to USB 2.0, available at http://www.usb.org/developers/devclass_docs

3 Abbreviations

APDU	Application Protocol Data Unit
ACA	Auto Contingent Allegiance
CBW	Command Block Wrapper
CBWCB	CBW Command Block
CSW	Command Status Wrapper
ICCD	Integrated Circuit(s) Card Devices
CDC	Communication Device Class
EEM	Ethernet Emulation Model
NACA	Normal ACA
SCSI	Small Computer System Interface
SCWS	Smart Card Web Server
SWP	Single Wire Protocol

4 Definitions

Integrated Circuit Card	The most general term for a smart card is "ICC". It is always a physical and logical entity either a SIM or a UICC.
Card Session	The period starting with the electrical activation of the card and ending with its electrical deactivation.
ISO Interface	The UICC to terminal interface specified by ISO 7816-3 and further detailed by ETSI TS 102 221[3].
USB Interface	The UICC to terminal interface specified by ETSI TS 102 600[6].

5 Hardware Requirements

The following chapter describes the new hardware requirements resulting from an introduction of the USB interface to a UICC and a terminal.

5.1 Physical Characteristics

5.1.1 Form Factor

The physical characteristics of the USB UICC are as defined in TS 102 221. USB UICC adopts the form factor physical type plug-in as specified in UICC specification TS 102 221.

5.1.2 Contact Assignment

The USB UICC makes use of four contacts: VCC (C1), GND (C5), IC_DP (C4) and IC_DM (C8). VCC and GND are reused from the existing ISO interface defined in 7816 parts 2 [12] and 3 [13]. The IC_DP and IC_DM contacts have been specifically added for Inter-Chip USB I/O [21] and adapted to the UICC – Terminal interface in ETSI TS 102 600 [6].

The ISO Interface contacts RST (C2) and CLK (C3) do not affect the USB interface.

The SWP contact (C6) may provide the means to communicate to an NFC device according to the UICC-CLF interface specification TS 102 613 [7]. This contact also does not affect the USB interface.

Figure 1 shows the UICC pin layout (bottom view).

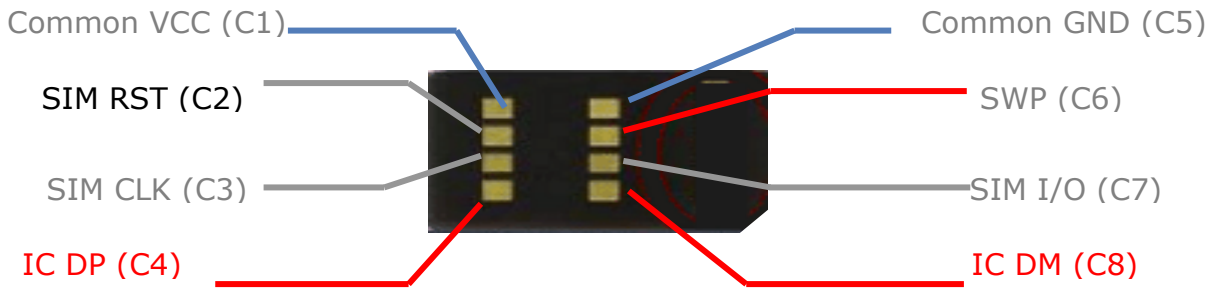


Figure 1 - (U) SIM Physical interface with USB and legacy ISO/IEC 7816 interfaces

5.1.3 Card Connector

The ISO 7816-2/3 standard interface can be used without any changes to access (U)SIM part of the USB UICC card or for use with legacy (U)SIM cards.

The handset must integrate an 8-pin connector compliant with the ISO/IEC 7816-2 standard.

The USB UICC card may be semi rigid and therefore less flexible than standard SIMs. Special care should be taken when designing the connector to avoid or limit torque and bend constraints on the SIM module. The connector should provide means to prevent torque and bend as much as possible. For example, a connector with two positions can meet this requirement: an open position allowing an easy card insertion without any torsion and a lock position or special slot to insert the SIM card.

5.2 Electrical Characteristics

5.2.1 Power Supply

The power signals, IC_VDD and GND, are respectively applied on the VCC (C1) and GND (C5) contacts. These contacts may be shared by several interfaces such as 102 221, ETSI 102 600 and ETSI 102 613.

The interfaces are not linked and it is possible e.g. to communicate over the USB interface without any impact on the SWP interface. But, there is one exception to this independence of the interfaces: When the terminal requires turning off the power or needs to perform a cold reset (which inherently requires to turn off and on again the power supply) the status of all interfaces will be affected.

5.2.1.1 Power Consumption

As long as the UICC has not detected a USB capable terminal the UICC shall behave according to the ISO Interface [3] specification. Once a USB capable terminal has been detected and no power negotiation has successfully completed a maximum UICC power consumption of 10 mA applies according to TS 102 600 [6].
When USB UICC interface is selected and a successful power negotiation procedure has been performed by the terminal and the UICC, the USB enabled terminal shall supply the negotiated current. The UICC shall not exceed the negotiated supply current.

The terminal should be capable of sourcing a current which is sufficient to operate UICCs implementing a Flash based Mass Storage.

To counteract spikes it is recommended that handsets place a capacitor of sufficient value as close as possible to the contacting elements to the power supply pad (between C1 and C5). While the capacitance has to be high enough to ensure stable voltage levels even through Flash programming cycles of the UICC it should be low enough to not interfere with the SWP interface when operated in battery-off mode.

The current to be provided should reflect the characteristics of the current Flash technology. Up-to-date Flash memory technology relevant to the UICC is implemented in voltage class B for higher memory sizes.

UICCs are expected to deal with this restriction in two ways: during power negotiation they may indicate to the USB terminal that a class B activation is preferred or they may ask for a class C' support which transforms the reduction in voltage into an increase in current consumption.

Terminal manufacturers not supporting class B or the higher power consumption for class C' should be aware that they may exclude important UICC functionality from their support.

Flash Memories technology is well known in the market and also to the handset manufacturers. Many handset models from various brands are using Flash memory technology integrated in the handset implementing mass storage applications. These handsets are known to comply to the power supply requirements stated e.g. for the support of MMC and SD cards.

5.2.1.2 Signals Voltage Level Class

The electrical signals voltage levels should be of the same voltage class as defined for contact C1 (VDD).

Signal levels from higher voltage than the selected voltage class for contact C1 shall not be applied.

For example if class C' supply voltage is selected, the signals from voltage level of class B should not be supported by the UICC or terminal, or vice versa.

5.2.1.3 Interaction with Other Interfaces

When the terminal is switched off or the battery is down or low, the terminal may provide voltage on contact C1 and C5, for example when using SWP interface in low power mode. In this mode the current consumption of the UICC is limited and USB interface is unlikely to be supported.

If the UICC is in SWP low power mode and the terminal detects that the battery is available, the terminal may select working in voltage class C. If the terminal would like to operate at voltage class B, it should power down and reinitiate the power negotiation procedure.

5.3 USB Attach

The ETSI 102 600 USB attach process may be interpreted in a different way than defined in IC_USB standard. The ETSI 102 600 defines that the peripheral may "attach" at any time after C4 and C8 are maintained in state L for at least 10 ms or the condition described in the procedure using ATR is met (i.e. a special PPS command is sent).

In order to comply with other protocols, the terminal may delay the assertion of C4 and C8 to L state after power up. Once the C4 and C8 are maintained in state L for at least 10 ms, the USB UICC supporting ETSI 102 600 should attach according the IC_USB time limits.

6 Software Requirements

Three USB classes have been defined by ETSI [6] to be used for the communication between a USB UICC and a USB enabled terminal:

- APDU over USB (ICCD)
- Mass Storage and
- CDC EEM

The following chapters describe the software requirements related to these classes.

6.1 USB Classes

Coding:

Note that data words are transmitted on the USB interface in the little endian format.

6.1 APDU over USB (ICCD)

6.1.1 Relevant Specifications

The USB Smartcard Device Class specification covers the transport of APDUs over USB using a choice of transfer modes. The SmartCard Class has a unique USB descriptor, however there are 2 flavors, namely CCID (Chip Card Interface Device) intended for ISO 7816-3 readers using USB, and ICCD (Integrated Circuit Card Device) optimized for USB SmartCards. The later one is the reference for USB UICCs. The ISO 7816-12 international standard also provides useful tips for implementing the USB Smartcard device Class.

APDU transfer in USB Bulk mode is common to ICCD and CCID devices, though ICCD devices do not use the CCID reader specific requests. On the other hand, only ICCD devices may make use of Control Transfer, which comes in 2 versions:

- Type A control transfer, which was based on T=0, allows easy porting of existing smartcard stacks at the cost of important protocol overhead
- Type B control transfer is more similar to what is used for bulk transfer.

The use of an interrupt pipe is optional in the USB Smartcard Class.

6.1.2 Support of legacy applications over USB

All USB UICCs and USB UICC-enabled terminals shall support APDU-level encapsulation over Version B Control transfer with no Interrupt pipe for short APDUs. Mapping of APDUs into TPDUs does not take place when transferring APDUs over USB, it is the APDUs rather than TPDUs that are transmitted.

This allows the support of APDU commands and existing applications like USIM or ISIM over the USB interface. The commands shall then be processed by the UICC application as if they were coming from the ISO interface.

Using Control Transfer rather than Bulk Transfer is of interest for Authentication commands as they need to be prioritized over other traffic and Control mode uses a dedicated part of the bandwidth while bulk transfer has the lowest priority. However, an additional configuration using APDU-level encapsulation over a dedicated pair of bulk pipes and (no interrupt pipe) may be requested for USB UICC-enabled terminals and USB UICC by applications relying on APDU communication to exchange large amount of data.

The ICCD Class specification has been written under the assumption that this class was the only one existing on the smart card. This explains that two class specific commands have been defined which (seem to) go beyond the scope of this class and affect the overall card state:

Control Tranfer	**Bulk Transfer**	**Description (*)**
ICC_POWER_ON	PC_to_RDR_IccPowerOn	Exits the initial state of a USB-ICC. The ATR is returned in the data stage of the subsequent DATA_BLOCK request.
ICC_POWER_OFF	PC_to_RDR_IccPowerOff	Sets the USB-ICC to initial conditions.

ICCD Class specific requests [19]

(*) Note that the "Description" quotes ISO 7816-12 and reflects the misunderstanding of the ICCD Class specification when using the term "USB-ICC" instead of referring to the "USB-ICCD Class".

In the case of the USB UICC where more than one class may be active any interference between the classes shall be avoided. For this reason TS 102 600[6] states that

- the PPS procedure does not apply when transferring APDUs over USB.
- cold and warm resets are logically performed by USB commands and
- processing time extension may be requested as defined in the USB Integrated Circuit Card Device (ICCD) Class specification.

The role of the Warm Reset used to reinitialise the complete UICC when the ISO interface is used is taken over by the USB Reset in the case where the terminal and the UICC communicate over the USB interface.

6.1.3 USB UICC Proactivity

All applications and features based on TS 102 221, such as the Card Application Toolkit defined in TS 102 223, may be used in the context of APDU communication over USB. The existing polling mechanism defined in TS 102 223 to provide proactivity remains available to ease compatibility with older toolkit applications, however the default polling interval is extended in the case of USB to limit the power consumption.

When the USB UICC indicates support of remote wakeup in its configuration using this smartcard functional interface, remote wakeup should be activated by the USB UICC-enabled terminals whenever possible as an alternative to using the polling mechanism specified in TS 102 223 for UICC proactivity.

After a remote wake up, the terminal shall send a STATUS command on this functional interface to allow the UICC to start a proactive session.

6.2 Mass Storage

The implementation of the USB Mass Storage Class inside the UICC provides a large and secure storage for all kinds of multimedia data of relevance to the user. This data may be organized by the user herself or by the operator e.g. in the case of device customization or service related information.

Beyond this "directly" visible memory, which is presented to the user in the form of a file system by the mobile phone, parts of the mass storage are likely to be managed internally by the UICC and not accessible through the USB Mass Storage interface. This storage will be used to support applications like the next generation multimedia phone book and the Smart Card Web Server. Although this part of the mass storage is outside the scope of the information given in this section it shall be noted that its presence is closely linked to the availability of the USB Mass Storage interface and that it requires the same (electrical) hardware support as described above.

6.2.1 Relevant Specifications

The ETSI USB specification [6] requires the support of the following standards for an USB Mass Storage implementation:

- Mass Storage Specification Overview [17]
- Mass Storage Bulk Only 1.0 specification [16]
- SCSI Primary Commands - 3 [9]
- SCSI Block Commands - 2 [10]

The "Mass Storage Specification Overview"
provides an overview of all the specifications that describe how Mass Storage devices behave on the USB bus. The specification referenced here and to be supported on the UICC-handset interface is the Mass Storage Bulk Only specification [16].

In addition the specification defines the descriptor values for the Mass Storage device subclass (bInterfaceSubclass) and the transport protocol (bInterfaceProtocol) values to be used in the enumeration phase. For the USB UICC the following values apply:

Descriptor Element	Value	Description
bInterfaceClass	'08'	Mass Storage
bInterfaceSubclass	'06'	SCSI Transparent Command Set
bInterfaceProtocol	'50'	Bulk-Only Transport

The "Mass Storage Bulk Only" specification
describes the transport of Mass Storage commands over bulk endpoints only (as opposed to using bulk plus interrupt endpoints). This includes the specification of the

standard class descriptors, a Mass Storage command wrapping layer and of two class specific commands.

The "SCSI Primary Commands" and "SCSI Block Commands" specifications describe the specific commands to manage a Mass Storage, basically to store and retrieve data.

6.2.2 Enumeration

During enumeration the class (*bInterfaceClass)*, subclass (*bInterfaceSubclass*) and protocol (*bInterfaceProtocol*) values listed above have to be used by the USB UICC to offer a Mass Storage Device implementing the SCSI Transparent Command Set over a Bulk-Only interface.

This is typically not done by coding the values in the Device descriptor (where the respective data elements therefore are set to zero), even if only a single USB class is supported by the UICC, but through the Interface descriptor.

For the Mass Storage Class a single Interface descriptor and no *bAlternateSetting* exists. For an example of an Interface descriptor coding see 6.4.1.2

The Bulk-Only Mass Storage Class requires one Bulk IN and one Bulk Out endpoint. For performance reasons a packet size (*wMaxPacketSize*) of 64 bytes should be supported by the USB UICC and the USB Terminal.

6.2.3 Mass Storage Commands

Bulk-Only Mass Storage commands consist of class specific commands, transport commands and the actual Mass storage management commands. The class specific commands and the transport commands make up a transport layer for transmitting the Mass Storage management commands, which in the case of the USB UICC and terminal consist of SCSI commands.

6.2.3.1 Class Specific Commands

A USB UICC and terminal must support two class specific Mass Storage commands which are transported over the default control pipe:

- Bulk-Only Mass Storage Reset
- Get Max LUN

Any other class specific request sent over the Mass Storage interface must be responded with a Request Error (STALL).

6.2.3.1.1 Bulk-Only Mass Storage Reset

This class specific request resets the Mass Storage device and its associated interface, i.e. it prepares the device to receive the next transport command from the host.

Note:
To achieve a resetting of the data toggle bits and the STALL conditions the Reset request has to be followed by a CLEAR_FEATURE (endpoint halt) to both, the Bulk-In and Bulk-Out endpoint.

6.2.3.1.2 GET MAX LUN request

According to the USB Mass Storage specification[16] a device may implement several Logical Mass Storage Units (LUN) which share the same basic device characteristics. The LUN is then used by the host to address the relevant unit (in the bCBWLUN parameter) when sending transport commands.

A USB UICC may implement more than one Logical Unit, e.g. to split the storage into a protected and an unprotected part. Therefore a USB Terminal shall support at least two LUNs.

Although a device may STALL this request if only a single LUN is supported the USB UICC always answers the request.

6.2.3.2 USB Transport Commands

The USB Mass Storage interface consists of two protocol layers, the USB transport layer, specified by the USB Implementers Forum, and the storage management command layer which consists of SCSI commands standardised by the "T10" organisation. Both layers define their own protocol frames, command and error handling.

On the USB layer SCSI commands are transferred between a host and a device in the form of requests and responses made up of two or three USB transfers:

- **Command Transport** (OUT): the host sends a command using a Command Block Wrapper (CBW).
- **Data IN/OUT**: The command related (optional) input or output data is transferred to the device or host
- **Status Transport** (IN): the device returns the command status using a Command Status Wrapper (CSW).

Command Block Wrapper (CBW)

The CBW field ***bCBWLUN*** contains the Logical Unit Number (LUN) addressed by the command (cf. GET MAX LUN request above).

Command Status Wrapper (CSW)

The ***dCSWStatus*** field of the CSW reports the result of the command execution and contains one of the following values:

- '00': Command Passed
- '01': Command Failed (CBW is valid but processing failed)
- '02': Phase Error

The values '03' to 'FF' are reserved.

Phase Error
This error code indicates that the results of processing further CBWs will be indeterminate until the device is reset.

Error Recovery
The interface must respond to invalid CBWs by STALLING all traffic until Reset Recovery.

The interface must respond properly to Get_Status requests at all times.

Host Error Handling
If the host receives a CSW which is not valid, then the host shall perform a Reset Recovery. If the host receives a CSW which is not meaningful, then the host *may* perform a Reset Recovery.

Thirteen Cases Assertions

Because the CBW contains redundant information related to the data transfer direction (*bmCBWFlags* and *dCBWDataTranferLength* fields) which have their counterparts in the command block (CB), inconsistencies in the command coding may arise known as the Thirteen Cases.

The behaviour of the UICC and the terminal regarding the Thirteen Cases shall be as defined in chapter 6 of the USB Mass Storage specification [16].

6.2.4 SCSI Storage Management Commands

The USB Mass Storage transport command wrapper conveys SCSI commands and responses between the UICC and the Terminal. The following command set must be supported by the UICC and the terminal:

Command	Op-Code	UICC	Terminal	Specification
TEST UNIT READY	'00'	M	O	SPC-3
REQUEST SENSE	'03'	M	O	SPC-3
FORMAT UNIT	'04'	M (1)	X	SBC-2
INQUIRY	'12'	M	M	SPC-3
SEND DIAGNOSTICS	'1D'	M (1)	X	SPC-3
READ CAPACITY (10)	'25'	M	M	SBC-2
READ(10)	'28'	M	M	SBC-2
REPORT LUNs	'A0'	M	M	SPC-3
WRITE (10)	'2A'	M	M	SBC-2
MODE SENSE (6)	1A	M	M	SPC-3
M = Mandatory (1) Minimal functional support provided O = Optional X = not recommended				

Direct Access Device Commands [10]

Command Coding

Note that data words contained in SCSI commands are transmitted in the big endian format.

Command Descriptor Block (CDB)

The Command Descriptor Block specifies a command and its parameter. It consists of an Operation Code, command specific parameters and a Control field. The Control field of all commands is structured as follows:

CDB Control Byte

Bit	7	6	5	4	3	2	1	0
	Vendor specific		Reserved			NACA	Obsolete	Obsolete

A USB terminal shall not expect a USB UICC to support neither

- *Vendor specific* parameters nor
- *NACA* (Normal ACA).

If the terminal sends a command with a Control field value different from zero it shall be prepared to receive a CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

SCSI Command Set Implementation

The SCSI commands shall be implemented as specified in the respective standards. Those commands which only support a subset of the specified options or should be handled in a specific way are further detailed in the following.

6.2.4.1 INQUIRY Command

The INQUIRY allows an application client to determine the configuration of a Mass Storage device.

EVPD (Enable Vital Product Data):
The terminal shall not expect any Vital Product Data to be supported by the UICC.

Page Code:
The UICC shall (at least) support the retrieval of the standard configuration data when the page code is set to zero.

INQUIRY Response

A USB UICC must support the request for the page zero information with the following data:

INQUIRY Response – Standard Data

Bit / Byte	7	6	5	4	3	2	1	0
0	Peripheral Qualifier: 000			Peripheral Device Type 0 0000				
1	RMB: 1	Reserved (000 0000)						
2	Version ('05' = SPC-3)							
3	Obsolete (00)		NormACA	HiSup	Response Data Format			
4	ADDITIONAL LENGTH (length-5 = n-4)							
5	SCCS	ACC	TPGS		3PC	Reserved		Protect
6	BQUE	EncServ	VS	MultiP	MChngr	Obsolete		[ADDR16]
7	Obsolete		[Wbus16]	[SYNC]	LINKED	Obsol.	CmdQue	VS
8-15	(MSB) T10 Vendor Identification (LSB)							
16-31	(MSB) Product Identification (LSB)							
32-35	(MSB) Product Level (LSB)							
36-55	(MSB) Vendor Specific (LSB)							
56	Reserved							
57	Reserved							
58-59	(MSB) Version Descriptor 1 (LSB)							
	...							
72-73	(MSB) Version Descriptor 8 (LSB)							
74-95	Reserved							
96 – n	Vendor specific							

"[<flags>]" - These fields are only relevant to SCSC parallel interface devices.

A USB terminal should support at least the following USB UICC configuration:

Attribute	Value	Description
Peripheral Qualifier	000b	Peripheral device connected to this unit
Peripheral Device Type	0 0000b	Direct access block device
RMB (Removable Medium	1b	Removable
Version	'05'	Device complies to SPC-3
NormACA (Normal ACA)	0b	The UICC does not support the NACA bit set to one in the CDB CONTROL byte and does not support the ACA task attribute
HiSup (Hierarchical Support)	?b	If more than one LUN is supported by the USB UICC the hierarchical addressing model for LUN assignment shall be indicated as supported (1b). If only a single LUN is supported the HiSup bit is set to zero.

Response Data Format	'2'	The data format conforms to this standard (i.e. SPC-3)
Additional Length	<variable >	The number of bytes send by the device excluding bytes 0-4
SCCS	0b	No Embedded Storage Array Controller Component supported
ACC	0b	No Access Controls Coordinator contained
TPGS	00b	No Target Port Group Support (asymmetric access)
3PC	0b	No 3rd Part Copy (extended copy commands) supported
Protect	0b	No Protection Information supported
BQUE	0b	No Task Management combined with CmdQue supported
ENCSERV	0b	No Enclosure Services supported
VS	0b	Vendor Specific
MultiP	0b	No Multi Port support
Mchanger	0b	No Media Changer commands supported
CmdQue	0b	No Task Management supported (see BQUE)
T10 VENDOR IDENTIFICATION	<variable >	Eight bytes of left-aligned ASCII characters identifying the product vendor. The T10 vendor identification shall be one assigned by INCITS. A list of assigned T10 vendor identifications is in Annex E and on the T10 web site (http://www.t10.org). NOTE 21 – The T10 web site (http://www.t10.org) provides a convenient means to request an identification code.
PRODUCT IDENTIFICATION	<variable >	Sixteen bytes of left-aligned ASCII characters defined by the vendor.
PRODUCT REVISION LEVEL	<variable >	Four bytes of left-aligned ASCII characters defined by the vendor.

According to SPC-3 [9] a USB UICC has to return all information described above up to and including the PRODUCT REVISION LEVEL. A USB Terminal shall not expect a USB UICC to return any other values.

6.2.4.2 FORMAT UNIT

The FORMAT UNIT command has been designed to allow a host to format a storage at a low level by creating a logical view of the physical structure of the device. This process includes the definition of the size and the number of sectors and the creation of control structures, e.g. to prevent defect blocks from being used.

The FORMAT UNIT command does not perform a higher level structuring of the device like the generation of a file system (like FAT) which is established with help of the WRITE command.

For this reason USB Terminals shall not send this command to the UICC.

For the reason of formal compatibility with the SPC-3 specification USB UICCs is only required to implement this command with minimal feature support, i.e. it may terminate the command only successfully if the command parameters are identical with the existing format or return an error code of the form:

SCSI Sense Key	Meaning
0x5	Illegal request

SCSI Additional Sense Code	SCSI Qualifier Code	Meaning
0x20	0x00	Invalid command operation code

6.2.4.3 SEND DIAGNOSTICS

The SEND DIAGNOSTICS command allows a host to control the operational state of the device, in particular to check for (physical) defects of memory sectors.

Because USB UICCs incorporate internal mechanisms to only show operational sectors to a host (by separating the physical view of the memory from the logical view) this command cannot provide valuable information to a host.

Therefore USB Terminals shall not send this command to the UICC.

For the reason of formal compatibility with the SPC-3 specification USB UICCs is only required to implement this command with minimal feature support or respond with the following error:

SCSI Sense Key	Meaning
0x5	Illegal request

SCSI Additional Sense Code	SCSI Qualifier Code	Meaning
0x20	0x00	Invalid command operation code

MODE SENSE (6)

The MODE SENSE command shall be supported by a UICC to allow the terminal to determine if the storage of a LUN is write protected. To retrieve this information the terminal shall send the command with the *Page Code* parameter set to *SBC_PAGE_RETURN_ALL* ('3F'). The terminal shall not expect any *Block Descriptors* or *Mode Pages* to be returned.

6.2.5 Mass Storage Formatting

The terminal shall support Mass Storage formats according to [15] with the following additions.

The terminal should support:
- single LUN with single partition without Master Boot Record (MBR)
- single LUN with single partition with MBR
- two (or more) LUNs where each LUN can be formatted as listed above

The terminal shall be prepared to handle partitions correctly which are write protected. In the particular case where the UICC offers two partitions or LUNs it is likely that one of them is write protected.

In the case where a UICC presents two partitions within a single LUN with one being write protected this should be the second partition.

A typical use case for presenting two LUNs with a different write protect status is the provisioning of marketing data to the user in the case where the operator (partially) sponsors the memory and wants to prevent the user from deleting this data while another area (LUN) can be controlled by the user for her own purposes.

6.3 Communication

The implementation of the Communication Device Class Ethernet Emulation Model class (CDC EEM) will provide the USB UICC with a mean to transport Ethernet frames across the USB bus. This new class shall be seen as a new device subclass intended to be used with the USB Class Definition for Communication Devices Specification (USB CDC).

The USB CDC EEM interface Class is used to provide IP connectivity to the UICC. The handset and the UICC device constitute a network segment and so they are able to communicate directly.

6.3.1 Relevant specification

The ETSI USB specification [6] requires the support of the following standards for a CDC EEM Class implementation:

- Universal Serial Buss Communication Class Subclass Specification for Ethernet Emulation Model Devices [20]

6.3.2 Enumeration

During enumeration the following values have to be indicated by the UICC to identify the CDC EEM Class:

Descriptor Element	**Value**	**Description**
bInterfaceClass	'02'	Communication Device Class
bInterfaceSubclass	'0C'	Communication Device Subclass EEM
bInterfaceProtocol	'07'	EEM

The values are typically not coded in the Device descriptor (where the respective data elements therefore are set to zero) even if only a single USB class is supported by the UICC but through the Interface descriptor.

For the CDC EEM Class a single Interface descriptor and no *bAlternateSetting* exist.

For an example of an interface coding see 6.4.1.2

The CDC EEM Class requires one Bulk IN and one Bulk Out endpoint. For performance reasons a packet size (*wMaxPacketSize*) of 64 byte should be supported by the USB UICC and the USB Terminal.

6.3.3 CDC EEM Commands

According to CDC EEM[20] and ETSI 102 600[6] the following CDC EEM command set must be supported by the UICC and the terminal:

Command	Op-Code	UICC	Terminal	Specification
ECHO	'00'	M	M	CDC EEM 1.0
ECHO RESPONSE	'01'	M	M	CDC EEM 1.0
SUSPEND HINT	'02'	M	M	CDC EEM 1.0
RESPONSE HINT	'03'	M	M	CDC EEM 1.0
RESPONSE COMPLETE HINT	'04'	M	M	CDC EEM 1.0
TICKLE	'05'	O	O	CDC EEM 1.0
M = Mandatory O = Optional				

CDC EEM Commands [20]

6.3.4 SUSPEND AND RESUME

For the CDC EEM 1.0 interface, "SuspendHint" and "remote wakeup" capabilities are specified as optional. The card supports "SuspendHint" and may support "remote wakeup". The handset operating system shall support "SuspendHint" and "RemoteWakeup".

A CDC EEM command "SuspendHint" is sent by the device to inform the host that the device has entered a state where it is safe to suspend. The optional "remote wakeup" could be necessary for a suspended card requiring, for example, client connection to external world when triggered in conjunction with a communication over the SWP. In this case, the device will ask the USB host for a remote wake up.

When receiving "SuspendHint" command from the device and if it has no remaining data to transfer, the EEM driver MUST suspend the interface.

6.3.5 IP connectivity

The ETSI specification TS 102.483 [5] defines how an Internet Protocol connection may be established between a UICC and a terminal connected through a UICC-Terminal Interface able to carry Internet Protocol packets.

The UICC will be able to act as a combination of the following basic configurations:

- A client of a server located on the terminal
- A server for a client located on the terminal
- A client of a server located in a network reachable through the terminal
- A server for a client located in a network reachable through the terminal

Depending on the final applications, the actual configuration may be a combination of these basic configurations.

6.3.5.1 Internet Protocol Configuration:

The handset and the USB UICC constitute a network segment and are so able to communicate directly to each other without consuming or even needing other network resources as e.g. a GPRS connection.

However, for a different purpose the USB UICC may require TCP/IP access to remote nodes on the network, and remote TCP/IP nodes on the network might send packets to be routed to the card.

The protocols to be supported by a USB UICC and a USB terminal include

- IP V6 (Internet Protocol Version 6)
- IP V4 (Internet Protocol Version 4)
- TCP (Transport Control Protocol)
- UDP (User Datagram Protocol)

The IPv4/IPv6 inter-working is defined in ETSI TS 102.483 [5].

Although the standard requires the support of both IPv4 and IPv6, CDC EEM implementations should not be postponed because of a late availability of IPv6 infrastructures.

6.3.5.2 Logical names convention

- The UICC name for terminal is **localuicc**.
- The terminal name for UICC is **localterminal.**

The "localuicc" is used by applications as the smart card name. For instance, the handset browser can send HTTP requests to the smart card embedded HTTP server. A handset Domain Name Resolution service SHALL map "localuicc" to the IP address of the single UICC.

The "localterminal" name may be used by handset applications to refer to handset local services. A handset Domain Name Resolution service SHOULD map the "localterminal" name to the IP address of the terminal in the card sub network.

The card will use the static address to address the handset applications; for instance the card may access a handset service to setup the Access Point Number (APN) to be used to open a connection from the card to Internet.

6.3.5.3 Routing, Network Address Translation and port forwarding

UICC applications may require a specific APN to be used by the terminal when forwarding UICC originating IP messages to a remote host. So far no method for informing the terminal about the APN to be used has been defined by ETSI (or any other standardization organization).

6.3.5.4 Ethernet address configuration

Message exchange over an Ethernet requires an allocation of MAC addresses.

Handset Ethernet MAC address

Depending on handset manufacturer requirements, the terminal shall either use a globally unique MAC address or a locally administered MAC address but with a different setting in the most significant byte. Any other value than 0x82, which is reserved for the USB UICC, can be used.

USBUICC Ethernet MAC address

The USB UICC MAC address is allocated by the UICC manufacturer using a derived locally administered method. The MAC address is of the form 82-xx-xx-xx-xx-xx, where xx-xx-xx-xx-xx are 5 bytes derived from a unique value on the smart card, e.g. a truncated SHA1 hash of the ICCID (serial number digits of the SIM card).

6.3.6 Use of CDC EEM versus BIP

The Bearer Independent Protocol (BIP)[4] defines a set of APDUs allowing the UICC to perform IP communication based on the handset protocol stack. With the introduction of CDC EEM which moves the IP stack into the UICC two access paths to UICC applications exist. To avoid any inconsistent application states only one of the protocols shall be used to exchange messages between two applications during a card session. To promote the introduction of the CDC EEM this protocol should be given priority.

6.4 USB Classes Interworking

This paragraph gives some guidance on handling USB UICC configurations which consist of more than one USB class.

6.4.1 Composite Device Configuration

A Composite Device Configuration describes the case where a single physical USB device offers two or more device classes to a host.

Device Descriptor
The Device Descriptor is identical in all configuration cases because the class related information has to be stored in other descriptors. The parameters *bDeviceClass*, *bDeviceSubClass*, *bDeviceProtocol* therefore have to be set to zero.

Configuration Descriptor
The only parameter in the Device Descriptor depending on the classes supported is the number of interfaces (*bNumInterfaces*) which has to be adapted according to number of interfaces supported (starting with value One).

According the USB IC specification [21] the bMaxPower field in the Configuration Descriptor shall contain a value of 4 or less. This does not indicate that the USB UICC requires a power of 8 mA or less. Rather the power supply requirement is defined by ETSI TS 102 600 as follows:

„Under all operating conditions, a USB UICC-enabled terminal shall be able to supply at least 10 mA until either a power negotiation occurs on the IC USB interface or the TS 102 221 interface is selected."[3]

Interface Descriptor
An example of the Interface descriptors of a composite configuration is given below.

Note:
USB standard also allows for the indication of multiple interfaces through an Interface Association Descriptor (IAD) [22]. As this descriptor type has been designed for the case where the device interfaces should be addressed by a single device function driver it should not be used by a USB UICC.

Endpoint Descriptor

Endpoint descriptors only exist for Bulk and Interrupt endpoints with the latter not being used on the UICC – terminal interface. The only relevant characteristic of the Bulk endpoints defined in the descriptor is the maximum size of the related buffer which should indicate the highest value possible for a Full Speed interface, namely 64 bytes.

To allow all classes, APDU, Mass Storage and CDC EEM, to be supported with an acceptable performance two separate endpoints per class, i.e. six Bulk endpoints in total shall be supported by the UICC and the terminal.

6.4.1.1 APDU Interfaces

In the case where the ICCD class is only supported through the control pipe the descriptors are coded as described in [6].

When the USB UICC supports both ICCD over control and bulk transfer, the bulk transfer option should be indicated through an additional interface descriptor with the *bAlternateSetting* value set to One.

The ICCD Class requires an additional specific descriptor for the Smart Card Device Class (*bDescriptorType* = '21') to be send by the UICC. Although the USB specification does not require such a descriptor to be put into any specific position within the sequence of descriptors it is recommended to place it between the interface and endpoint descriptors of the ICCD Class.

6.4.1.2 APDU, Mass Storage and Communication Interface

In the case where more than one USB class is supported by the USB UICC the parameters of the respective protocols are provided in standard interface descriptors.

For the case where two APDU interfaces, the Mass Storage and the CDC/EEM communication interfaces are supported by the USB UICC an example coding of the respective interface descriptors is given below.

Offset	Field	Size	Value	Description
0	*blength*	1	'09'	Size of this descriptor in bytes.
1	*bDescriptorType*	1	'04'	INTERFACE
2	*bInterfaceNumber*	1	'00'	First interface (zero-based value) in the array of concurrent interfaces supported by this configuration.
3	*bAlternateSetting*	1	'00'	Value used to select Alternate Setting for the interface identified in the prior field. No Alternate Setting are supported for BOT.
4	*bNumEndpoints*	1	'00'	Only control pipe for Control Transfer version B
5	*bInterfaceClass*	1	'0B'	ICCD
6	*bInterfaceSubClass*	1	'00'	No subclass
7	*bInterfaceProtocol*	1	'02'	Control transfer version B
8	*iInterface*	1	'00'	Index of String Descriptor describing this interface (no descriptor available in this case)

APDU Interface Descriptor: ICCD control transfer version B

Offset	Field	Size	Value	Description
0	*bLength*	1	'09'	Size of this descriptor in bytes.
1	*bDescriptorType*	1	'04'	INTERFACE
2	*bInterfaceNumber*	1	'00'	First interface (zero-based value)
3	*bAlternateSetting*	1	'01'	Value used to select Alternate Setting (here: Bulk transfer) for the interface identified in the prior field (i.e. the ICCD interface).
4	*bNumEndpoints*	1	'02'	Two Bulk endpoints
5	*bInterfaceClass*	1	'0B'	ICCD
6	*bInterfaceSubClass*	1	'00'	No subclass
7	*bInterfaceProtocol*	1	'00'	Bulk transfer
8	*iInterface*	1	'00'	Index of String Descriptor describing this interface (no descriptor available in this case)

Alternate APDU Interface Descriptor: ICCD Bulk transfer

Offset	Field	Size	Value	Description
0	*bLength*	1	'09'	Size of this descriptor in bytes.
1	*bDescriptorType*	1	'04'	INTERFACE
2	*bInterfaceNumber*	1	'01'	Second interface (zero-based)
3	*bAlternateSetting*	1	'00'	No Alternate Setting are supported for BOT.
4	*bNumEndpoints*	1	'02'	Number of endpoints (excluding endpoint zero) used USB BOT: 1 IN + 1 OUT Bulk endpoint
5	*bInterfaceClass*	1	'08'	Mass Storage Class
6	*bInterfaceSubClass*	1	'06'	SCSI Transparent Command Set
7	*bInterfaceProtocol*	1	'50'	Bulk-Only Transport Protocol
8	*iInterface*	1	'00'	Index of String Descriptor describing this interface (no descriptor available in this case)

Mass Storage Interface Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	'09'	Size of this descriptor in bytes.
1	*bDescriptorType*	1	'04'	INTERFACE
2	*bInterfaceNumber*	1	'02'	Third interface (zero-based)
3	*bAlternateSetting*	1	'00'	No Alternate Setting is supported for BOT.
4	*bNumEndpoints*	1	'02'	Number of endpoints (excluding endpoint zero) used USB BOT: 1 IN + 1 OUT Bulk endpoint
5	*bInterfaceClass*	1	'02'	Communication Device Class
6	*bInterfaceSubClass*	1	'0C'	Communication Device Subclass EEM
7	*bInterfaceProtocol*	1	'07'	EEM
8	*iInterface*	1	'00'	Index of String Descriptor describing this interface (no descriptor available in this case)

CDC EEM Interface Descriptor

A Parallel ISO / USB Interface Activation

This section describes some of the specific aspects to be taken into consideration when developing a boot driver which has to be able to handle both, the activation of the ISO interface and the activation of the new USB interface.

Because a UICC and a Terminal may or may not support the new USB interface the following configurations can be detected when a Terminal activates a UICC:

Terminal supporting / UICC supporting	ETSI 102 221 only interface	ETSI 102 221 and ETSI 102 600 interface
ETSI 102 221 only	Only ETSI 102 221 can be used	Only ETSI 102 221 can be used
ETSI 102 221 and ETSI 102 600	Only ETSI 102 221 can be used	ETSI 102 221 or ETSI 102 600 can be used

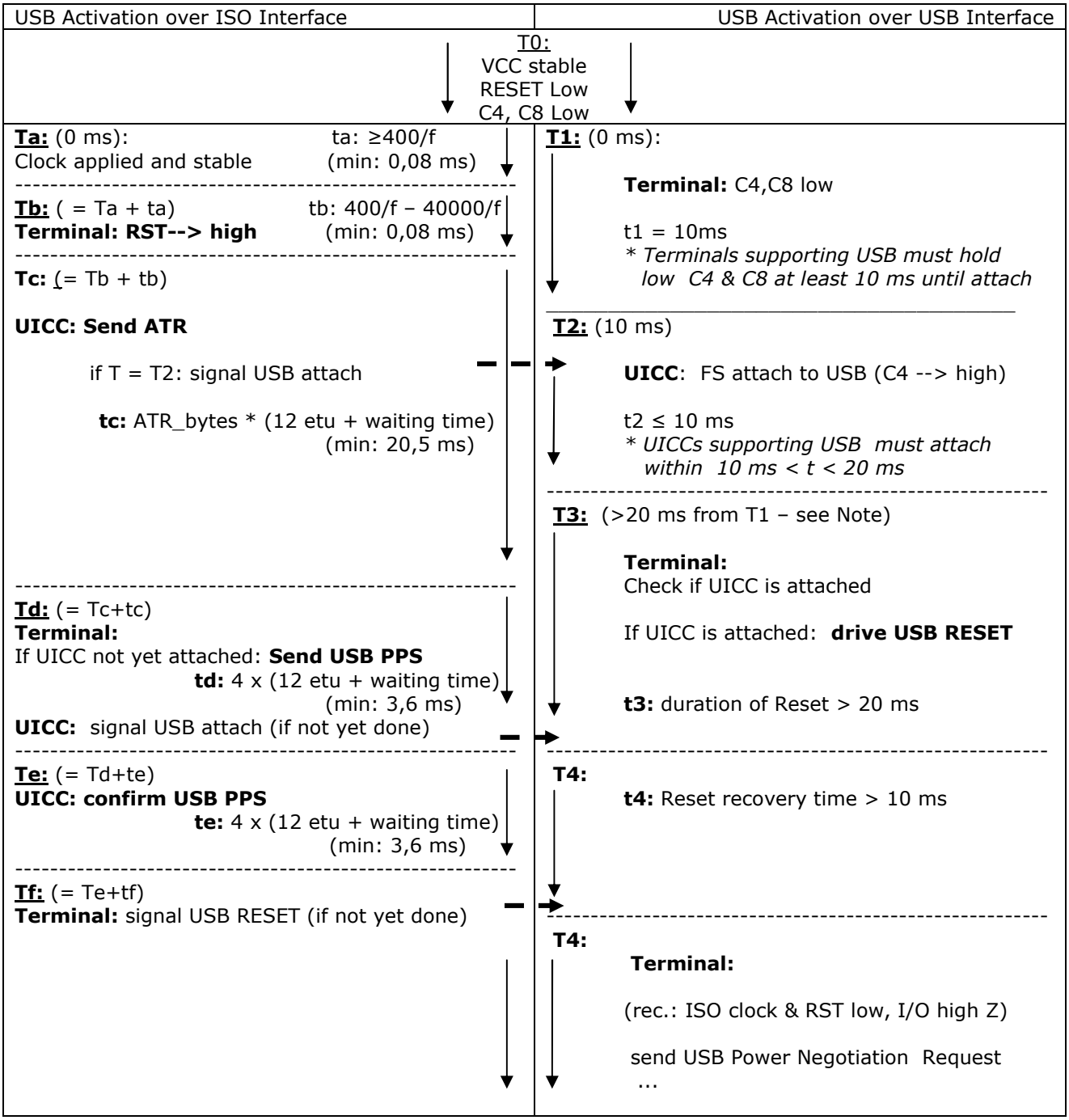
When the terminal only supports the ETSI 102 221 ("ISO") interface obviously nothing is changed. When the terminal supports both interfaces a choice can be made to try the activation of the USB interface

- over the ISO interface
- over the USB interface, or
- over both interfaces in parallel

Because a UICC may not support the USB interface starting the activation on the USB interface only may delay the activation of the ISO interface by the time the UICC is given to attach to the USB bus, which is 20 ms from a stable power-on state.

To start the USB interface activation on the ISO interface only on the other hand may delay the availability of the USB interface because an ISO interface activation may take quite long compared to an USB interface activation.

As visible from the activation timing table below these delays can be prevented when a parallel activation on both interfaces is performed.



Terminal and UICC process USB activation on ISO and USB Interface in parallel

Note
According to the USB Inter-Chip specification a host shall not check for the attachment of a peripheral before time Δth1 with a minimum of 20 ms has expired. The time is calculated from the point where a stable power is supplied (see [21], chapter 6.3 for the definition of Δth1 and figure 6-2 for the description]. This means that T3 is expected to be typically close to but never less than 20ms.

It should be noted that the legacy terminals may hold C4 and C8 low during activation. Therefore only the USB Reset signals the support of the USB interface by the terminal to the UICC.

While the duration of an USB activation over the USB interface only is quite static the ISO interface activation may vary significantly depending on the guard / work time consumed by the card and the clock frequency applied to the card by the terminal. To allow for a fast availability of the UICC services, cards in the market typically stay far below the maximum allowed values.

Time intervals	Description	Time in ms(*)
ta min.	Clk stable & RST low	0,08
tb min. tb typical tbmax.	1st bit of ATR sent by UICC after RST high	0,08 3,00 40,00
tc min. tc typical tc max	last bit of ATR sent by UICC	20,50 60,00 114.397,44
td/te min. td/te typical td/te max.	last bit of PPS request sent by terminal	3,57 5,49 38.440
td/te min. td/te typical td/te max.	last bit of PPS response sent by UICC	3,57 5,49 38.440

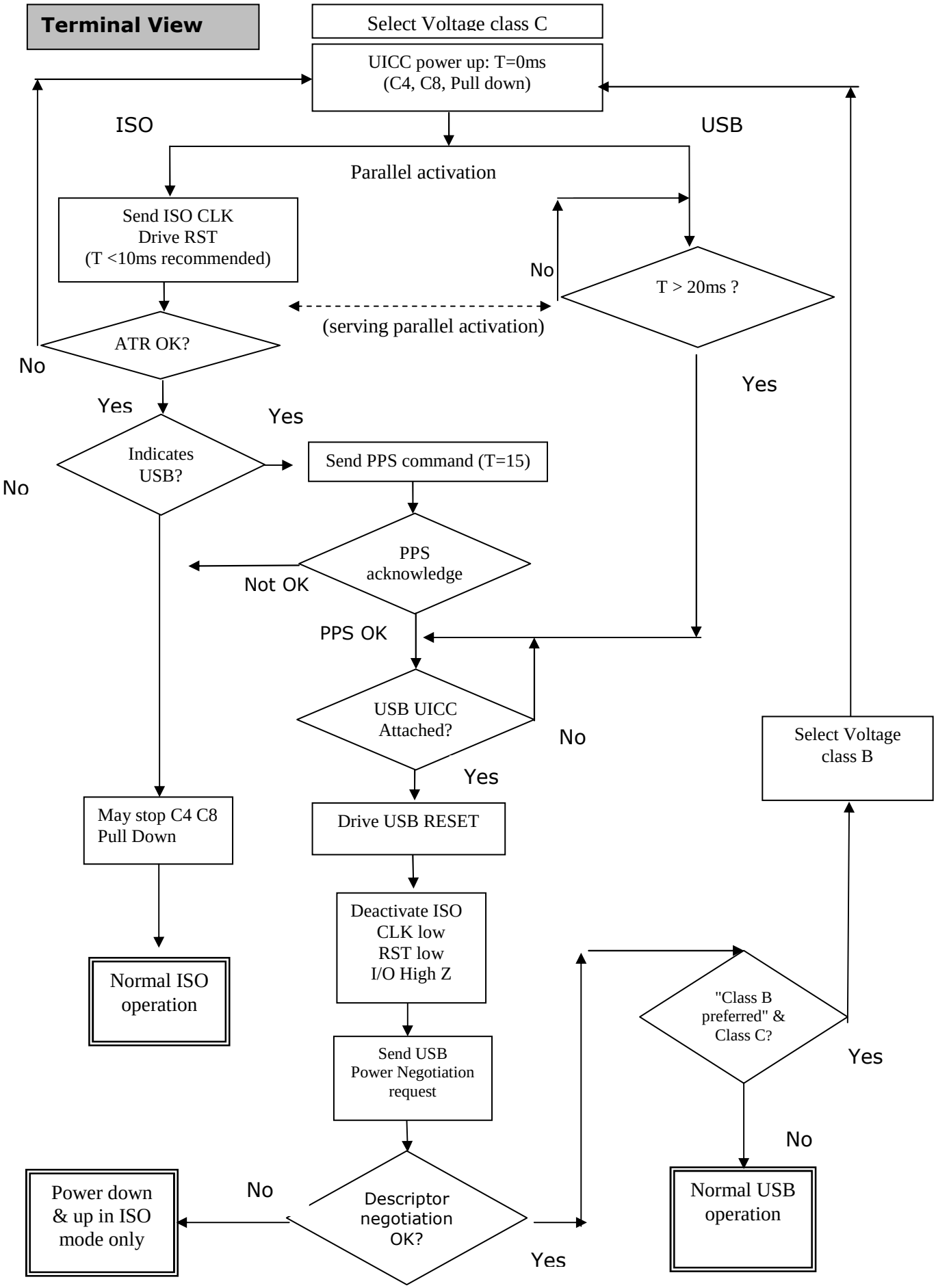
ISO Interface Activation Timing

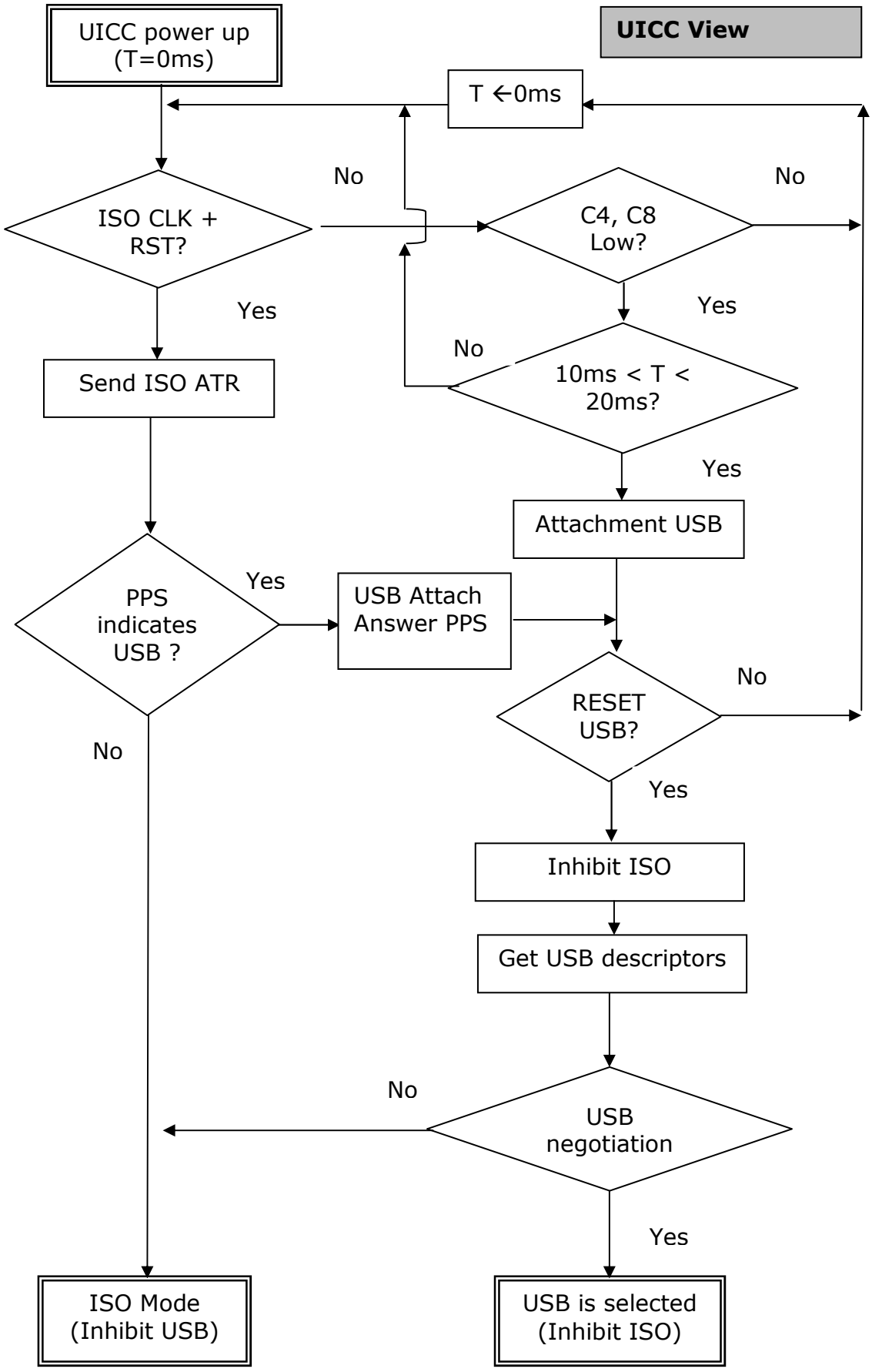
(*) Conditions

1 etu (bit time)	372 clock cycles
1 character	10 etu (1 start bit, 8 character bits, 1 parity bit)
t<x> min:	5 MHz Character transmission: guard time / waiting time = 12 ETU ATR: 23 bytes (assumption for coding all important information) PPS: 4 bytes
t<x>typical:	3,25 MHz Character transmission: guard time / waiting time = 12 ETU ATR: 23 bytes, extra guard time after first character PPS: 4 bytes
t<x>max:	1 MHz Character transmission: guard time / waiting time =9610 ETU ATR: 32 bytes, extra guard time after first character PPS: 4 bytes

Because there will be many ISO and few USB UICCs in the market for quite some time, a parallel activation on both interfaces which avoids an increased ISO start-up for ISO only UICCs but still allows to save time when USB is supported will contribute to a faster availability of the whole mobile system.

In the following two flowcharts are provided to allow for a closer look into the decisions being made during the course of a parallel interface activation on the terminal and on the UICC side.


Terminal View
Select Voltage class C
UICC power up: T=0ms
(C4, C8, Pull down)
ISO
USB
Parallel activation
Send ISO CLK
Drive RST
(T <10ms recommended)
No
T > 20ms ?
(serving parallel activation)
ATR OK?
No
Yes
Yes
Yes
Indicates
USB?
No
Send PPS command (T=15)
PPS
acknowledge
Not OK
PPS OK
USB UICC
Attached?
No
Yes
Select Voltage
class B
May stop C4 C8
Pull Down
Drive USB RESET
Deactivate ISO
CLK low
RST low
I/O High Z
Normal ISO
operation
"Class B
preferred" &
Class C?
Yes
Send USB
Power Negotiation
request
No
Power down
& up in ISO
mode only
No
Descriptor
negotiation
OK?
Yes
Normal USB
operation


UICC power up
(T=0ms)
UICC View
T ←0ms
No
No
ISO CLK +
RST?
C4, C8
Low?
Yes
Yes
No
Send ISO ATR
10ms < T <
20ms?
Yes
Attachment USB
PPS
indicates
USB ?
Yes
USB Attach
Answer PPS
No
RESET
USB?
No
Yes
Inhibit ISO
Get USB descriptors
No
USB
negotiation
Yes
ISO Mode
(Inhibit USB)
USB is selected
(Inhibit ISO)

B Use Cases (Annex)

This section describes some of the main use cases currently considered by mobile network operators to exploit the capabilities of the new USB UICC.

B.1 Handset & Operator Services Customisation

An important point for Mobile Network Operators is the ability to personalize and to better configure the services offered to their subscribers through the handset. Such personalization could be done at two levels:

- the customization of the handset (User Interface & handset applications according to Operator needs)
- Local HTTP portal (On-Card portal) to better promote Operator services

The main idea here is to leverage on the USB UICC card to provide the User Interface, the applications and a local HTTP portal enabling for example a better market segmentation, to allow remote change of the configuration (for example when the user is moving from one segment to another) and to enable the use of local services based on the USB UICC either directly from the User Interface or from the browser.

Detailed requirements need to be discussed directly with the Mobile Network Operators.

User Interface customization from USB UICC

Several levels of UI customizations are envisaged (only some icons, home screen, full UI customization).

The handset must have the possibility to:

- Retrieve UI configuration resources (both description and data) from the USB UICC at boot time.
- Change dynamically the UI configuration resources if requested (for example after an OTA update of the SIM related to a user segment change).
- Use a default & specified file (name to be defined) stored in the file system of the USB UICC to get all the information needed for the UI customization.

Application distribution from USB UICC

Handset applications (e.g. MNO specific applications like IM client, email client, MMS client …) will be stored in the USB UICC card memory. There should be a "MNO set-up" mechanism that will manage automatically the installation of those applications with minimum user interaction. An agent will launch the set-up, e.g. when a card is inserted for the first time. When completed, the handset will be customized with MNO applications.

Handset Browser

Using the handset browser for customisation would involve the following functions:

- WAP 2.0 XHTML/HTML multimode browser support including HTTP1.1, persistent connections and chunk encoding
- Support of local browsing from the USB UICC card
- Offline mode support
- Possibility to use local resources like web pages directly from the USB UICC card.

The Web Server Card can be seen as a local web server allowing the Mobile Network Operator to provide pre-loaded web sites for the different market segments in order to ease service discovery and content teasing. It also allows the end-user to personalize a portable web home page.

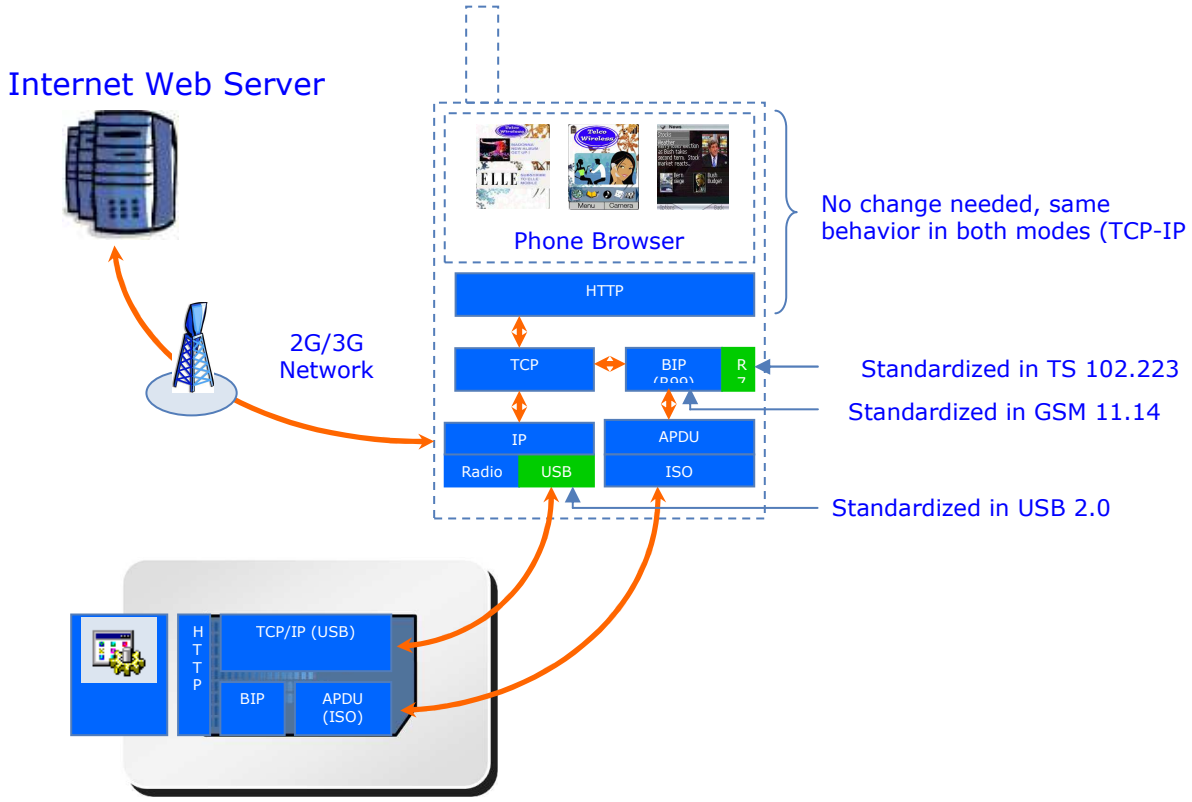


Figure 2 – *Web Server Card Overview*

B.2 Connectivity Framework

In addition to the standard APDU interface and USB mass storage, the USB UICC card will be able to support the TCP protocol and a "connectivity framework". This

framework will be used for different applications like the Web Server Card, Contact Management API (currently being specified by 3GPP CT6) or DRM applications.
The use cases that need to be addressed fall into the following categories:

- Enable local IP connectivity between a handset application and the USB UICC card. For example it will be used to allow the handset standard browser to connect to a web server located in the card
- Enable connectivity between the card and a network server. In this case the communication is initialized by the USB UICC card. It can be used for example to have the smart card checking for local portal update on the network side.
- Enable connectivity between a network server and the card. In this case the communication is initialized by the server. It can be used by the operator to launch a card update campaign or to download new services to the card.
- Enable connectivity between a PC and the USB UICC. In this case, the handset is connected with the PC via USB cable, or over Bluetooth, WiFi, etc. An example of use is to browse the card web server from the PC web browser.

Remote IP Connection

The architecture for an IP communication between the USB UICC and a remote server is described in the following block graphic.

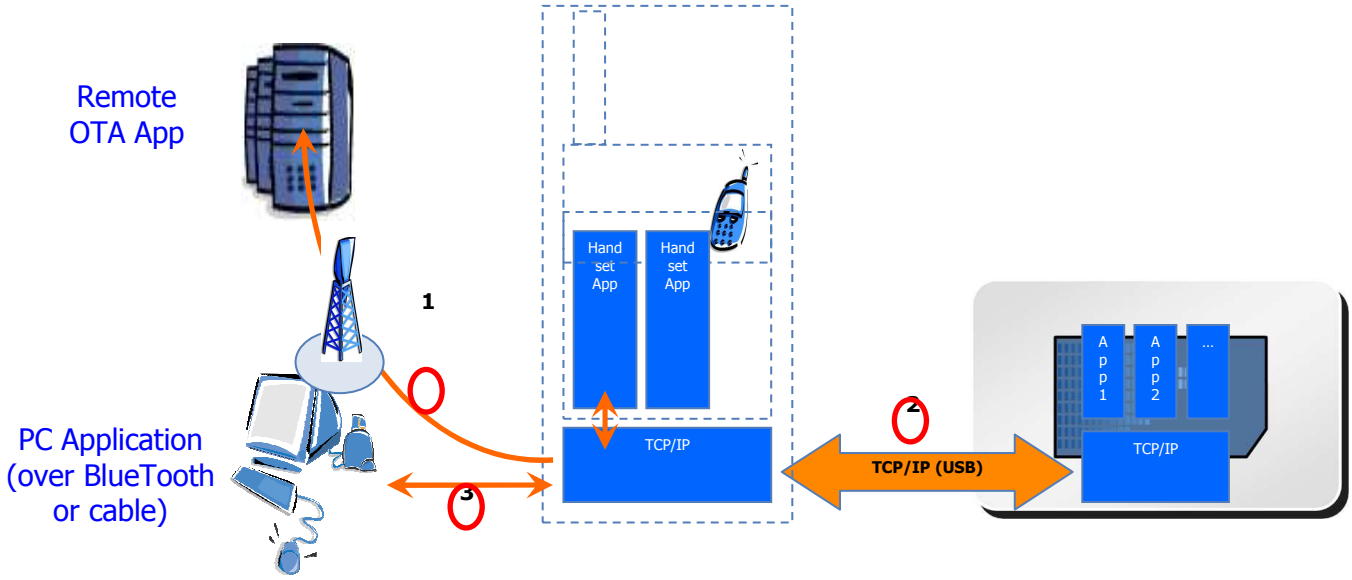


Figure 3 –*TCP/IP Framework*

On the handset side the requirements are the following ones:

- TCP/IP stack available
- Local multihoming requirement to allow the handset to manage access to a private network between the handset and the USB UICC card in addition to the 2G/3G network. (A multihomed host is a host that can be addressed with more than one IP address.)
- IP forwarding to allow exchanges of packets between the different IP networks.
- Network Address Translation to allow the USB UICC card to connect to external services. Network Address Translation (NAT) in the terminal solves the problem that the UICC address is not known to external hosts. The terminal maps the external address to the local UICC address when forwarding messages to the UICC and vice versa for messages to remote hosts.

B.3 Contact Management

The USB UICC card will be able to host personal data collected by the handset and to ensure synchronisation with other devices, in order to make user data accessible at any time. The kind of data to be synchronised are phonebook entries, tasks, meetings, calendar, photos, ringtones, videos, music, email, …

The synchronisation is based on the OMA-DS technology using the HTTP binding.

3GPP CT6 currently works on a Release 8 specification of a new Contact Management facility on the UICC and the mobile phone based on OMA DS. To allow for early deployment projects a detailed first step specification for the implementation of the OMA DS clients with OMA DS server cards is provided by the SIMAlliance.

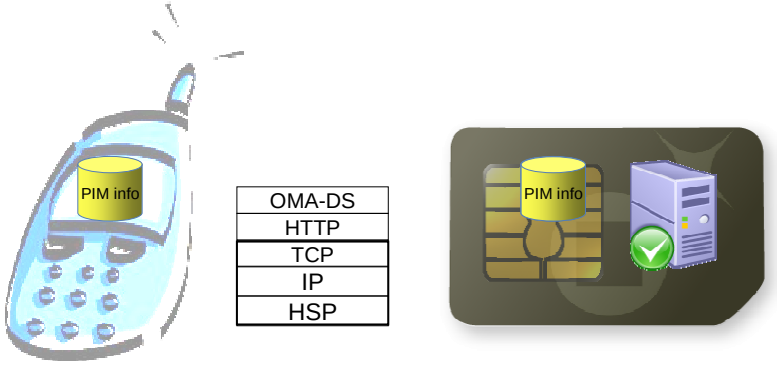


Figure 4 - OMA DS Client and Server

C Change History (Annex)

This annex lists all changes made to the present document since its initial approval.

Date	VERS	REL	SUBJECT	Resulting Version
19.06.08		7	Final draft version of document	0.9