Open Connection Manager API

Approved Version 1.0 – 26 Jan 2016

Open Mobile Alliance

OMA-TS-OpenCMAPI-V1_0-20160126-A

Contents

Figures

Tables

1. Scope

This specification of the OpenCMAPI defines an interface, through which connection management services are made available to different applications.

The specification addresses the requirements enumerated in **[OpenCMAPI-RD]** and adheres to the architecture described in **[OpenCMAPI-AD]**.

2. References

2.1 Normative References

[3GPP TR 21.905] "TR 21.905 Technical Specification Group Services and System Aspects; Vocabulary for 3GPP Specifications", 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/21_series/21.905/

[3GPP TS 22.002] "TS 22.00205 Technical Specification Group Services and System Aspects; Circuit Bearer Services (BS) supported by a Public Land Mobile Network (PLMN)", 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/22_series/22.002/

[3GPP TS 22.011] "TS 22.011 Technical Specification Group Services and System Aspects; Service accessibility", 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/22_series/22.011/

[3GPP TS 22.022] "TS 22.022 Technical Specification Group Services and System Aspects; Personalisation of Mobile Equipment (ME), Mobile functionality specification", 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/22_series/22.022/

[3GPP TS 22.030] "TS 22.030 Technical Specification Group Services and System Aspects; Man-Machine Interface (MMI) of the User Equipment (UE)", 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/22_series/22.030/

[3GPP TS 22.101] "TS 22.101 Technical Specification Group Services and System Aspects; Service aspects; Service principles", 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/22_series/22.101/

[3GPP TS 23.003] "TS 23.003 Technical Specification Group Services and System Aspects; Numbering, addressing and identification", 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/23_series/23.003/

[3GPP TS 23.038] "TS 23.038 Technical Specification Group Services and System Aspects; Alphabets and language-specific information", 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/23_series/23.038/

[3GPP TS 23.040] "TS 23.040 Technical Specification Group Services and System Aspects; Technical realization of the Short Message Service (SMS)", 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/23_series/23.040/

[3GPP TS 23.060] "TS 23.060 Technical Specification Group Services and System Aspects; General Packet Radio Service (GPRS); Service description; Stage 2", 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/23_series/23.060/

[3GPP TS 23.107] "TS 23.107 Technical Specification Group Services and System Aspects; Quality of Service (QoS) concept and architecture", 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/23_series/23.107/

[3GPP TS 24.008] "TS 24.008 Technical Specification Group Core Network and Terminals; Mobile radio interface Layer 3 specification; Core network protocols; Stage 3", 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/24_series/24.008/

[3GPP TS 24.090] "TS 24.090 Technical Specification Group Core Network and Terminals; Unstructured Supplementary Service Data (USSD)", 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/24_series/24.090/

[3GPP TS 25.323] "TS 25.323 Technical Specification Group Radio Access Network; Packet Data Convergence Protocol (PDCP) specification", 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/25_series/25.323/

[3GPP TS 27.007] "TS 27.007 Technical Specification Group Services and System Aspects; AT command set for User Equipment (UE)", 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/27_series/27.007/

[3GPP TS 31.101] "TS 31.101 Technical Specification Group Core Network and Terminals; UICC-terminal interface; Physical and logical characteristics, 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/31_series/31.101/

[3GPP TS 31.102] "TS 31.102 Technical Specification Smart Cards; Characteristics of the Universal Subscriber Identity Module (USIM) application", 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/31_series/31.102/

[3GPP TS 31.103] "TS 31.103 Technical Specification Group Core Network and Terminals; Characteristics of the IP Multimedia Services Identity Module (ISIM) application", 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/31_series/31.103/

[3GPP TS 31.111] "TS 31.111 Technical Specification Group Core Network and Terminals; Universal Subscriber Identity Module (USIM), Application Toolkit (USAT)", 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/31_series/31.111/

[3GPP TS 33.401] "TS 33.401 Technical Specification Group Services and System Aspects; 3GPP System Architecture Evolution (SAE); Security architecture", 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/33_series/33.401/

[3GPP TS 33.402] "TS 33.402 Technical Specification Group Services and System Aspects; System Architecture Evolution (SAE); Security aspects of non-3GPP accesses", 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/33_series/33.402/

[3GPP TS 44.065] "TS 44.065 Technical Specification Group Core Network and Terminals; Mobile Station (MS) - Serving GPRS Support Node (SGSN); Subnetwork Dependent Convergence Protocol (SNDCP)", 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/44_series/44.065/

[3GPP TS 51.011] "TS 51.011 Technical Specification Group Terminals; Specification of the Subscriber Identity Module-Mobile Equipment (SIM - ME) interface", 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/51_series/51.011/

[3GPP TS 51.014] "TS 51.014 Technical Specification Group Terminals; Specification of the SIM Application Toolkit for the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface (Release 4)", 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/51_series/51.014/

[3GPP2 C.S0016] "Over-the-Air Service Provisioning of Mobile Stations in Spread Spectrum Systems", 3rd Generation Partnership Project 2 (3GPP2), Technical Specification 3GPP2 C.S0016,
URL: http://www.3gpp2.org/

[3GPP2 C.S0023] "Removable User Identity Module for Spread Spectrum Systems", 3rd Generation Partnership Project 2 (3GPP2), Technical Specification 3GPP2 C.S0023,
URL: http://www.3gpp2.org/

[3GPP2 C.S0035] "CDMA Card Application Toolkit (CCAT)", 3rd Generation Partnership Project 2 (3GPP2), Technical Specification 3GPP2 C.S0035,
URL: http://www.3gpp2.org/

[3GPP2 C.S0065] "Cdma2000 Application on UICC for Spread Spectrum Systems", 3rd Generation Partnership Project 2 (3GPP2), Technical Specification 3GPP2 C.S0065,
URL: http://www.3gpp2.org/

[3GPP2 C.S0068] "ME Personalization for CDMA2000 Spread Spectrum Systems", 3rd Generation Partnership Project 2 (3GPP2), Technical Specification 3GPP2 C.S0068,
URL: http://www.3gpp2.org/

[DMClientAPIFw v1.0] "Enabler Release for OMA Device Management Client API framework", OMA-ER-DMClientAPIfw-V1_0, Open Mobile Alliance™,
URL: http://www.openmobilealliance.org/

[ETSI TR 102 216] "TR 102 216 Technical Report Smart Cards; Vocabulary for Smart Card Platform specifications", v3.0.0, European Telecommunications Standards Institute (ETSI),
URL: http://www.etsi.org

[ETSI TS 102 221] "TS 102 221 Technical Specification, Smart Cards; UICC-Terminal interface; Physical and logical characteristics", European Telecommunications Standards Institute (ETSI),
URL: http://www.etsi.org

[ETSI TS 102 223] "TS 102 223 Technical Specification, Smart Cards; Card Application Toolkit (CAT)", European Telecommunications Standards Institute (ETSI),
URL: http://www.etsi.org

[GP, SE Access Control] "GlobalPlatform Device Technology, Secure Element Access Control", GlobalPlatform™,
URL: http://www.globalplatform.org/specificationsdevice.asp

[OpenCMAPI-AD] "Open Connection Manager API Architecture", Open Mobile Alliance™, OMA-AD-OpenCMAPI-V1_0-20111101-C.doc,
URL: http://www.openmobilealliance.org/

[OpenCMAPI-RD] "Open CM API Requirements", Open Mobile Alliance™, OMA-RD-OpenCMAPI-V1_0-20111101-C.doc,
URL: http://www.openmobilealliance.org/

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997,
URL: http://www.ietf.org/rfc/rfc2119.txt

[RFC4234] "Augmented BNF for Syntax Specifications: ABNF", D. Crocker, Ed., P. Overell. October 2005,
URL: http://www.ietf.org/rfc/rfc4234.txt

[RFC4291] "IP Version 6 Addressing Architecture", R. Hinden, S. Deering, February 2006,
URL: http://www.ietf.org/rfc/rfc4291.txt

[RFC5952] "A Recommendation for IPv6 Address Text Representation", S. Kawamura, M. Kawashima, August 2010
URL: http://www.ietf.org/rfc/rfc5952.txt

[SCRRULES] "SCR Rules and Procedures", Open Mobile Alliance™, OMA-ORG-SCR_Rules_and_Procedures,
URL: http://www.openmobilealliance.org/

2.2 Informative References

[OMADICT] "Dictionary for OMA Specifications", Version 2.8, Open Mobile Alliance™,
OMA-ORG-Dictionary-V2_8, URL: http://www.openmobilealliance.org/

3. Terminology and Conventions

3.1 Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

All sections and appendixes, except “Scope” and “Introduction”, are normative, unless they are explicitly indicated to be informative.

3.2 Definitions

AID	Application IDentifier as defined in [ETSI TR 102 216] and specified in [ETSI TS 102 221].
Cloud Device	Device that needs to be connected and using online services to be fully functional.
Connection Manager Application	An entity or application that manages different network connections based on user profiles associated with these connections.
CSIM	A CDMA2000 Subscriber Identity Module is an application defined in [3GPP2 C.S0065] residing on the UICC to register services provided by 3GPP2 mobile networks with the appropriate security.
Device	A device in the context of OpenCMAPI is defined as a hardware unit which is exposed through a proprietary driver and containing at least one radio for the purpose of two way communication. A device could contain more than one radio and in this case is referred to as a multi-function device. Example: 3GPP2 and also Wi-Fi/WLAN
Dormant	Connection still active but no traffic on Tx and Rx. In 3GPP context, PDP context is established but no traffic.
ISIM	An IP Multimedia Services Identity Module is an application defined in [3GPP TS 31.103] residing in the memory of the UICC, providing IP service identification, authentication and ability to set up Multimedia IP Services.
M2M	Any other device with an embedded modem module using wireless network(s) to communicate with other devices or networks. It could be for example a module for an automotive system or an alarm system or even a consumer device such as a camera or a portable game device with embedded module.
Mobile Broadband Device	A datacard or USB modem or dongle that can be plugged in a laptop to assume data connectivity to cellular networks
NAA	Network Access Application as defined in [ETSI TR 102 216]. Examples of NAA on UICC: CSIM, ISIM, USIM.
Network Identifier	Network Identifier as specified in [3GPP TS 23.003].
Operator Identifier	Operator Identifier as specified in [3GPP TS 23.003].
Profile/User Profile/Connection Profile	The term Profile or User Profile or Connection Profile will be used to identify the information needed to establish a connection. There are two types of Connection Profiles: cellular profiles for connection to cellular and WLAN profiles for connection to WLAN
Push Service	A service utilizing PUSH delivery mechanism that enables the mobile device to receive data traffic initiated by a dedicated server.
QNC	Quick Net Connect is a 2G data technology for circuit-switched 2G wireless networks
R-UIM	A Removable User Identity Module is a standalone module defined in [3GPP2 C.S0023] to register services provided by 3GPP2 mobile networks with the appropriate security.
SIM	A Subscriber Identity Module is a standalone module defined in [3GPP TS 51.011] to register services provided by 2G mobile networks with the appropriate security.
UICC	As defined in [OMA-DICT] and whose interface is specified in [3GPP TS 31.101].
UIM	A User Identity Module is a module defined in [3GPP2 C.S0023] to register services provided by 3GPP2

	mobile networks with the appropriate security. The UIM can either be a removable UIM (R-UIM) or a non-removable UIM.
USIM	A Universal Subscriber Identity Module is an application defined in [3GPP TS 31.102] residing in the memory of the UICC to register services provided by 3GPP mobile networks with the appropriate security.
Wireless Router	A cellular network device that combines a router, switch and Wi-Fi access point (Wi-Fi base station) in one box. In the case of OpenCMAPI, the network to provide connectivity will be a cellular network. There could be two sorts of Wireless router: portable for nomadic usage or fixed for home usage in the case of Digital Dividend for example however in the document they will be considered as the same.

3.3 Abbreviations

3GPP	3rd Generation Partnership Project
3GPP2	3rd Generation Partnership Project 2
AAA	Authentication, Authorization and Accounting
ACCOLC	Access Overload Class
AID	Application Identifier
AKA	Authentication and Key Agreement
AN-AAA	Access Network AAA
API	Application Programming Interface
APN	Access Point Name
ARA-M	Access Rule Application Master
ARF	Access Rule Files
CDMA	Code Division Multiple Access
CHAP	Challenge Handshake Authentication Protocol
CM	Connection Manager
CSIM	CDMA2000 Subscriber Identity Module
DM	Device Management
DNS	Domain Name System
EAP	Extensible Authentication Protocol
EDGE	Enhanced Data rates for GSM Evolution
ERI	Enhanced Roaming Indicator
ESN	Electronic Serial Number
ETSI	European Telecommunications Standards Institute
e-UTRAN	evolved Universal Terrestrial Radio Access Network
GAN	Generic Access Network
GERAN	GSM EDGE Radio Access Network
GNSS	Global Navigation Satellite System
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile communications
HA	Home Agent
HSPA	High Speed Packet Access
ISIM	IP Multimedia Services Identity Module

LTE	Long Term Evolution
MAC	Media Access Control
MCC	Mobile Country Code
MDN	Mobile Directory Number
MEID	Mobile station Equipment Identifier
MIN	Mobile Identification Number
MMS	Multimedia Messaging Service
MN-AAA	Mobile Node AAA
MNC	Mobile Network Code
MN-HA	Mobile Node Home Agent
MSID	Mobile Station Identifier
MSISDN	Mobile Station International Subscriber Directory Number
NAA	Network Access Application
NDIS	Network Driver Interface Specification
NIA	Network-Initiated Alert
NMEA	National Marine Electronics Association
ODM	Original Device Manufacturer
OEM	Original Equipment Manufacturer
OMA	Open Mobile Alliance
OpenCMAPI	Open Connection Manager (CM) Application Programming Interface (API)
PAP	Password Authentication Protocol
PDN	Public Data Network
PIN	Personal Identification Number
PLMN	Public Land Mobile Network
PRI	Preferred Roaming Indicator
PRL	Preferred Roaming List
PSK	PreShared Key
PUK	Personal Unlocking Key also called UNBLOCK PIN.
QoS	Quality of Service
RAS	Remote Access Service
RAT	Radio Access Technologies
RFC	Request For Comments
RSSI	Received Signal Strength Indicator
RTN	Reset to factory defaults
R-UIM	Removable User Identity Module
SCI	Slot Cycle Index
SCM	Station Class Mark
SCP	Session Configuration Protocol
SID	System Identifier
SIM	Subscriber Identity Module

SMS	Short Message Service
SMS-C	Short Message Service Center
SN	Sequence Number
SPC	Service Programming Code
SSID	Service Set Identifier
UI	User Interface
UICC	Universal Integrated Circuit card
UIM	User Identity Module
UMA	Unlicensed Mobile Access
UMTS	Universal Mobile Telecommunications System
USIM	Universal Subscriber Identity Module
USSD	Unstructured Supplementary Service Data
UTRAN	Universal Terrestrial Radio Access Network
VPN	Virtual Private Network
WEP	Wired Equivalent Privacy
Wi-Fi	Wireless Fidelity
WiMAX	Worldwide Interoperability for Microwave Access
WISPr	Wireless Internet Service Provider roaming
WLAN	Wireless Local Area Network
WPA2	Wi-Fi Protected Access Version 2
WPS	Wireless Protected Setup
WWAN	Wireless Wide Area Network

4. Introduction

With the multiplicity of networks available and the need for more connectivity, there is a market demand for a standardized API to provide connection management functionalities which would facilitate development and integration of Connection Manager Applications as well as to provide more status information about the connection to any application using mobile data services.

The goal of the OMA OpenCMAPI is to facilitate the development of, or even the adaptation of existing, Connection Manager Applications to the mobile environment and to provide additional services such as Information Status to applications relying on connectivity to mobile networks.

In this context, the Technical Specification for the OpenCMAPI provides resource definitions, data structures elements and defines APIs related to the connection management aspects.

4.1 Version 1.0

Version 1.0 of the Open CM API specification addresses the following aspects through the different Interfaces:

- [CMAPI-1]
  - Security and concurrency control function, e.g. access control and authorization
  - Device Discovery & Device Handling
  - Device Services
  - Cellular Network Connection Management
  - PIN/PUK Management
  - Interaction with the UICC
  - WLAN connection management
  - Information Status handling
  - Statistics handling
  - GNSS handling
  - SMS&USSD management
  - Push Data service management
- [CMAPI-2]: Callbacks & Registration/Deregistration to receive callbacks

5. Mandatory –Optional functions

5.1 Optional Function(s)

If an API function is mentioned as Optional and not supported by the implementation of the OpenCMAPI**,** it shall at least support the call of the function and the dedicated generic return value.

If a parameter is mentioned as optional into a function or optional within a structure, this parameter SHALL be implemented and supported by the OpenCMAPI. It will be up to the application to provide this parameter when calling the function.

The application indicates to the OpenCMAPI that a parameter is not to be used (because optional),

- By passing a null value for the pointer parameters or structure.
- By passing a 0xFF value for the non pointer byte parameters
- By passing a 0xFFFF value for the non pointer word parameters
- By passing a 0xFFFFFFFF value for the non pointer dword parameters
- By passing a 0xFFFFFFFFFFFFFFFF value for the non pointer qword parameters
- By passing a 0x FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF value for the non pointer dqword parameters

5.2 Mandatory & Optional Function(s) per device type

The following table describes if a group of functions is mandatory or optional depending on the device type. Each group of functions is corresponding to the dedicated section of the Technical specification.

	Mobile Broadband Device	**Laptop**	**Wireless Router**	**M2M**	**Smartphone**	**Tablet**	**Cloud Devices**
API Management	**M**	**M**	**M**	**M**	**M**	**M**	**M**
Device Discovery APIs	**M**	**M**	**M**	**M**	**M**	**M**	**M**
Cellular Network Management APIs	**M**	**M**	**M**	**M**	**M**	**M**	**M**
Connection Management APIs	**M**	**M**	**M**	**M**	**M**	**M**	**M**
Network Management APIs	**M**	**M**	**M**	**M**	**M**	**M**	**M**
CDMA2000 APIs	**O**	**O**	**O**	**O**	**O**	**O**	**O**
Device Service APIs	**M**	**M**	**M**	**M**	**M**	**M**	**M**
PINs/PUKs Management APIs	**M**	**M**	**M**	**M**	**M**	**M**	**M**

UICC Management APIs	**O**	**O**	**O**	**O**	**M**	**M**	**O**
WLAN APIs	**O**	**M**	**O**	**O**	**M**	**M**	**M**
Statistics APIs	**M**	**M**	**M**	**M**	**M**	**M**	**M**
Information Status APIs	**M**	**M**	**M**	**M**	**M**	**M**	**M**
SMS Management APIs	**M**	**M**	**M**	**M**	**M**	**M**	**M**
USSD Management APIs	**M**	**M**	**M**	**M**	**M**	**M**	**M**
GNSS APIs	**O**	**O**	**O**	**O**	**O**	**O**	**O**
Data Push Service Management APIs	**O**	**O**	**O**	**O**	**M**	**M**	**O**
Callback APIs	**M**	**M**	**M**	**M**	**M**	**M**	**M**

Table 1: Mandatory/Optional group of functions per device type

M – Mandatory

O – Optional

6. Design Convention and data structure definitions

6.1 Design convention

Throughout the document, the following design convention and terms will be used to denote absolute sizes of memory:

- All memory is caller allocated. The API will never allocate memory and return it through a function call which needs to be cleaned up
- Data returned through callbacks is valid only for the duration of the call and never needs to be cleaned up by the API user.
- boolean type is 4 bytes. Zero means false, Any non-zero value means true.
- byte will be used to denote 8 bit data values,
- word will be used to denote 2 byte values,
- dword will be used to denote 2 word values,
- qword will be used to denote 2 dword values,
- byte parameter [256] will indicate a 256 bytes long parameters,
- UTF8 will be used to represent a buffer with UTF8 data and null terminating symbol. When the buffer is referenced in a structure or function it shall be referenced by a pointer and will appear as UTF8*.
- The API is responsible to convert all data strings received from the device into UTF8.
- For every parameter designated as “input” only, const should be applied.
- a function pointer (or function*) is as a variable containing the address of a function.
- All structure definitions within this specification will be finite in size.This will serve to allow the caller to allocate a single block of memory for each passed in parameter. Any variable length data (like UTF8 strings) will reside after the finite structure(s) in memory and a pointer will be used to indicate where UTF8 strings and other finite structures reside. Either the caller or callee will layout the structures in this memory, depending on if the values are input or output. The caller will layout the memory where there is some data input and the callee will be responsible to layout (or re-layout) the memory when the data is output (or input/output). In either output case, the callee will signal insufficient size with a return code and indicate the necessary minimum size with the corresponding size parameter.
- Structure fields should be aligned on a byte boundary (i.e. # pragma pack (push 1)).
- Little endian shall be used by the application.

6.2 Data Type Definitions

6.2.1 RadioType

Definition RadioType
This prototype defines an enumeration of radio types. The following enumeration will be used throughout this document to define which radio a function operates on.

RadioType	dword	The following radio types are supported: • 0x00000001: GSM • 0x00000002: WCDMA/UMTS • 0x00000004: CDMA • 0x00000008: EVDO • 0x00000010: TD_SCDMA • 0x00000020: LTE • 0x00000040: WLAN

6.2.2 RadioState

Definition RadioState
This prototype defines an enumeration of radio power states.

RadioState	dword	The following radio states are supported: • 0x00000001: Radio On (Full Power) • 0x00000002: Radio On (Power Saving) - Optional • 0x00000003: Radio Off (Device still powered on) • 0x00000004: Radio Off (Device Off including hardware switch)

6.2.3 RFInfoType

Definition RFInfoType
This type defines a structure representing the information of a single RF link.

Field Name	Type	Description
radio	RadioType	See RadioType definition
maxDataRateUL	dword	Maximum bit rate supported for uplink in bit/s. The maximum data rate is set by the currently used technology on the network and the capability of the device and is the maximum supported which the device reports.
maxDataRateDL	dword	Maximum bit rate supported for downlink in bit/s. The maximum data rate is set by the currently used technology on the network and the capability of the device and is the maximum supported which the device reports.
frequencyBand	UTF8*	Contains the frequency band of the radio. This MAY also contain a postfix qualifier where appropriate (EX: “900”, “1900 PCS”, “1800 DCS”)
channelNumberUL	UTF8*	Channel number in use for the up link. May be comma separated if necessary. This is traffic channel only and does not include the control channels if used.
channelNumberDL	UTF8*	Channel number in use for the down link. May be comma separated if necessary. This is traffic channel only and does not include the control channels if used.

6.2.4 PLMNIconType

Definition PLMNIconType
This prototype defines a structure which describes the information related to the PLMN icon

Field Name	Type	Description
PLMNIconQualifier	byte	See [3GPP TS 31.102] for details. – '01' = icon is self-explanatory, i.e. if displayed, it replaces the corresponding name in text format. – '02' = icon is not self-explanatory, i.e. if displayed, it shall be displayed together with the corresponding name in text format.
PLMNIconName	UTF8*	• Name of the file containing the Icon information when the Icon Link is provided by the SmartCard under an URI (see [3GPP TS 31.102]) (e.g. PLMNIconName = “spng.jpg”) or • “IMG:” concatenated with the “Image Instance Descriptor value” when the Icon information are described through an Image Instance Descriptor of the EF_{IMG} file and the corresponding image storage data file inside the Smart Card (see [3GPP TS 31.102]).
PLMNIconFileContent	byte*	Content of the file containing the Icon information. Null pointer if no Icon is available
PLMNIconFileContentsize	dword	Buffer size of PLMNIconFileContent

6.2.5 NetworkInfoType

Definition NetworkInfoType
This prototype defines a structure which describes the information related to the network / PLMN

Field Name	Type	Description
systemID	dword	The radio system either 3GPP or 3GPP2 to which the function apply when the device is a multi-mode device. • 0x00000000: 3GPP • 0x00000001: 3GPP2
PLMNName	UTF8*	The name of the PLMN according to 3GPP and/or 3GPP2 name resolution [3GPP TS 22.101].
PLMNID	UTF8*	The PLMNID corresponding to the PLMN Name. The PLMNID is coded as a decimal value on the form "MCCMNC".
PLMNIcon	PLMNIconType*	Optional - The PLMN Icon.
NetworkStatus	dword	Specifies the status of the network: • 0x00000000:Registered • 0x00000001: Available • 0x00000002: Forbidden
PreferredStatus	dword	Specifies if the Network is in the preferred PLMN list or not: • 0x00000001: Network is in the preferred PLMN list • 0x00000002: Network is NOT in the preferred PLMN list
radio	RadioType*	See RadioType definition

6.2.6 IPAddress

Definition IPAddress
This prototype defines a structure which describes an IP Address.

Field Name	Type	Description
addressType	dword	A flag to indicate the type of address: • 0x00000001: IPv4 • 0x00000002: IPv6 • 0x00000003: IPv4v6
address	UTF8*	The address formatted in compliance with RFC 5952 and RFC 4291. Null pointer if no IPaddress is available

6.2.7 QoSStructure

Definition QoSStructure
This defines the structure used to communicate QoS event information.

validFeatures	dword	Based on the different traffic classes various features in this method are valid/invalid. This parameter describes which values are valid. If the defined bit is not set it means the corresponding parameter is not used and should not be used for any purpose by the application. • 0x00000001: Traffic Class • 0x00000002: Maximum Bitrate • 0x00000004: Guaranteed Bitrate • 0x00000008: Delivery Order • 0x00000010: Maximum SDU Size • 0x00000020: SDU Format Information • 0x00000040: SDU Error Ratio • 0x00000080: Residual Bit Error Ratio • 0x00000100: Delivery of Erroneous SDUs • 0x00000200: Transfer Delay • 0x00000400: Traffic Handling Priority • 0x00000800: Allocation Retention Priority • 0x00001000: Source Statistics Descriptor • 0x00002000: Signaling Indication • 0x00004000: Priority Level • 0x00008000: Pre-emption Capability • 0x00010000: Pre-emption Vulnerability
trafficClass	dword	The traffic class defines the type of application for which the bearer service is optimized. • 0x00000000: Conversational • 0x00000001: Streaming • 0x00000002 Interactive • 0x00000003 Background
maximumBitRate	dword	Maximum bitrate in kbps.
guaranteedBitRate	dword	Guaranteed bitrate in kbps.
deliveryOrder	dword	Indicates if in-sequence delivery is provided • 0x00000000: Not provided • 0x00000001: Provided

maximumSDUSize	dword	The maximum SDU size for which the network will satisfy the negotiated QoS. In Octets.
SDUFormatInformation	dword	The list of possible exact sized of SDUs
SDUErrorRatio	dword	Indicates the fraction of SDUs lost or detected as erroneous.
residualBitErrorRatio	dword	Indicates the undetected bit error ratio in the delivered SDUs
deliveryOfErroneousSDUs	dword	Indicates whether SDUs detected as erroneous shall be delivered or discarded. • 0x00000000: To be Discarded • 0x00000001: To be Delivered • 0x00000002: Detection is not used
transferDelay	dword	Indicates maximum delay for 95th percentile of the distribution of delay for all delivered SDUs during the lifetime of a bearer service (reported in milliseconds).
trafficHandlingPriority	dword	Defines the relative importance for handling of all SDUs belonging to the bearer compared to the SDUs of other bearers
allocationRetentionPriority	dword	Defines the relative importance compared to other bearers for allocation and retention of the bearer.
sourceStatisticsDescriptor	dword	Defines the characteristics of the source of submitted SDUs • 0x00000000: Speech • 0x00000001: Unknown
signallingIndication	dword	Defines the signaling nature of the submitted SDUs. • 0x00000000: No • 0x00000001: Yes
priorityLevel	dword	The Evolved Allocation/Retention Priority Level
preemptionCapability	dword	The Evolved Allocation/Retention Pre-emption Capability • 0x00000000: No • 0x00000001: Yes
preemptionVulnerability	dword	The Evolved Allocation/Retention Pre-emption Vulnerability • 0x00000000: No • 0x00000001: Yes

6.2.8 TrafficFlowTemplateType

Definition TrafficFlowTemplateType
This prototype defines a structure which describes a Traffic Flow Template for Packet Filtering. The following parameters are defined in [3GPP TS 23.060]. Some of the listed attributes may coexist in a Packet Filter while others mutually exclude each other, the possible combinations are shown in [3GPP TS 23.060]

Field Name	Type	Description
PacketFilterIdentifier	byte	A numeric parameter, value range from 1 to 16.
EvaluationPrecedenceIndex	byte	A numeric parameter. The value range is from 0 to 255.
SourceAddressandSubnetMask	UTF8*	The string is given as dot-separated numeric (0-255) parameters on the form: "a1.a2.a3.a4.m1.m2.m3.m4" for IPv4 or "a1.a2.a3.a4.a5.a6.a7.a8.a9.a10.a11.a12.a13.a14.a15.a16.m1.m2.m3.m4.m5.m6.m7.m8.m9.m10.m11.m12.m13.m14.m15.m16", for IPv6.
ProtocolNumber_NextHeader	byte	Protocol number for IPv4 / Next header for IPv6. A numeric parameter, value range from 0 to 255.
DestinationPortRange	UTF8*	The string is given as dot-separated numeric (0-65535) parameters on the form "f.t".
SourcePortRange	UTF8*	The string is given as dot-separated numeric (0-65535) parameters on the form "f.t".
IpsecSecurityParameterIndex	dword	Numeric value in hexadecimal format. The value range is from 0x00000000 to 0xFFFFFFFF.
TypeofServiceandMask_TrafficClassandMask	UTF8*	Type of service for IPv4 and mask Traffic class for IPv6 and mask The string is given as dot-separated numeric (0-255) parameters on the form "t.m".
FlowLabel	dword	Flow label for IPv6. Numeric value in hexadecimal format. The value range is from 0x00000000 to 0xFFFFFFFF. Valid for IPv6 only.
Direction	byte	A numeric parameter which specifies the transmission direction in which the packet filter shall be applied. 0 Pre-Release 7 TFT filter (see [3GPP TS 24.008], table 10.5.162) 1 Uplink 2 Downlink 3 Birectional (Up & Downlink).

6.2.9 SecondaryContextType

Definition SecondaryContextType
This prototype defines a structure which describes the QoS, the Data and Header compression and the TFT Packet Filter parameters for each Secondary Context.

Field Name	Type	Description
ContextStatus	byte	The status of the Secondary Context • 0x00 : not activated • 0x01 : activation/creation in progress • 0x02 : activated/created
RequestedQoS	QoSStructure*	Optional - Requested QoS for this Secondary Context.
MinimumQos	QoSStructure*	Optional - Minimum acceptable QoS for this Secondary Context
TFT	TrafficFlowTemplateType*	Traffic Flow Template indicating the parameters values to be used for Packet Filtering in this Secondary Context.
DataCompression	byte	A numeric parameter that controls PDP data compression for Primary Context (applicable for SNDCP only) (refer to [3GPP TS 44.065]). Possible values defined in [3GPP TS 27.007].
HeaderCompression	byte	A numeric parameter that controls PDP header compression (refer to [3GPP TS 44.065] and [3GPP TS 25.323]). Possible values defined in [3GPP TS 27.007].

6.2.10 CellularProfileType

Definition CellularProfileType
This prototype defines a structure which describes a Cellular Profile Type

Field Name	Type	Description
CellularProfileID	dword	The identification number of the Cellular Profile
CellularProfileName	UTF8*	The name of the Cellular Profile
UserName	UTF8*	The user name associated to the APN
Password	UTF8*	The password associated with the APN
PDP Type	dword	The type of PDP (Packet Data Protocol): • 0x00000001: IP • 0x00000002: PPP - PS data over GPRS or UMTS (PS connection with PDP type PPP)
APN	UTF8*	The APN used for this connection
Address	IPAddress	The IP address
PrimaryDNS	IPAddress	The primary DNS
SecondaryDNS	IPAddress	The secondary DNS

AuthType	dword	The Authentication Protocol type: • 0x00000000: CHAP only • 0x00000001: PAP only • 0x00000002: Automatic
UseDhcpForIP	boolean	Use DHCP for IP address. If this is true, then the IP field is unused.
UseDhcpForDNS	boolean	Use DHCP for DNS address. If this is true, then the PrimaryDNS and SecondaryDNS fields are unused.
TimeoutSeconds	dword	The time out in seconds
WINSPreferred	IPAddress*	Optional - The preferred WINS (Windows Internet Naming Service)
WINSAlternated	IPAddress*	Optional - The alternated WINS (Windows Internet Naming Service)
ServingPLMNs	UTF8*	Optional - List of possible serving PLMNs (MCCMNC numerical values separated by a coma and a space “, ”) on which the profile can be used (i.e; MCCMNCvalue1, MCCMNCvalue2, ..., MCCMNCvaluen). If the list is empty then the CellularProfile is valid for any PLMN. The check is done at the API level.
PCRequestedQoS	QoSStructure*	Optional - Requested QoS for Primary Context
PCMinimumQos	QoSStructure*	Optional - Minimum acceptable QoS for Primary Context
PCTFT	TrafficFlowTemplateType*	Optional - Traffic Flow Template indicating the parameters values to be used for Packet Filtering in the Primary Context.
PCDataCompression	byte	Optional - A numeric parameter that controls PDP data compression for Primary Context (applicable for SNDCP only) (refer to [3GPP TS 44.065]). Possible values defined in [3GPP TS 27.007].
PCHeaderCompression	byte	Optional - A numeric parameter that controls PDP header compression (refer to [3GPP TS 44.065] and [3GPP TS 25.323]). Possible values defined in [3GPP TS 27.007].
SecondaryContext1	SecondaryContextType*	Optional - 1st Secondary Context (if a null pointer value then no 1st SecondaryContext)
SecondaryContext2	SecondaryContextType*	Optional - 2nd Secondary Context (if a null pointer value then no 2nd SecondaryContext)
SecondaryContext3	SecondaryContextType*	Optional - 3rd Secondary Context (if a null pointer value then no 3rd SecondaryContext)
SecondaryContext4	SecondaryContextType*	Optional - 4th Secondary Context (if a null pointer value then no 4th SecondaryContext)
SecondaryContext5	SecondaryContextType*	Optional - 5th Secondary Context (if a null pointer value then no 5th SecondaryContext)
SecondaryContext6	SecondaryContextType*	Optional - 6th Secondary Context (if a null pointer value then no 6th SecondaryContext)

SecondaryContext7	SecondaryContext Type*	Optional - 7th Secondary Context (if a null pointer value then no 7th SecondaryContext)
SecondaryContext8	SecondaryContext Type*	Optional - 8th Secondary Context (if a null pointer value then no 8th SecondaryContext)
SecondaryContext9	SecondaryContext Type*	Optional - 9th Secondary Context (if a null pointer value then no 9th SecondaryContext)
SecondaryContext10	SecondaryContext Type*	Optional - 10th Secondary Context (if a null pointer value then no 10th SecondaryContext)
SecondaryContext11	SecondaryContext Type*	Optional - 11th Secondary Context (if a null pointer value then no 11th SecondaryContext)
SecondaryContext12	SecondaryContext Type*	Optional - 12th Secondary Context (if a null pointer value then no 12ve SecondaryContext)
SecondaryContext13	SecondaryContext Type*	Optional - 13th Secondary Context (if a null pointer value then no 13th SecondaryContext)
SecondaryContext14	SecondaryContext Type*	Optional - 14th Secondary Context (if a null pointer value then no 14th SecondaryContext)
SecondaryContext15	SecondaryContext Type*	Optional - 15th Secondary Context (if a null pointer value then no 15th SecondaryContext)
SecondaryContext16	SecondaryContext Type*	Optional - 16th Secondary Context (if a null pointer value then no 16th SecondaryContext)

6.2.11 WLANEncryptionType

Definition WLANEncryptionType
This prototype defines an enumeration of Encryption types for WLAN.

WLANEncryptionType	dword	The following encryption types are supported: • 0x00000001: none • 0x00000002: WEP • 0x00000004: TKIP • 0x00000008: AES/CCMP

6.2.12 WLANSecurityType

Definition WLANSecurityType
This prototype defines an enumeration of security types for WLAN.

WLANSecurityType	dword	The following security types are supported: • 0x00000001: Open (no security) • 0x00000002: WEP • 0x00000004: WPA • 0x00000008: WPA2 • 0x00000010: WPA_ENTERPRISE • 0x00000020: WPA2_ENTERPRISE

6.2.13 WLANNetwork

Definition WLANNetwork
This prototype defines a structure which describes a WLAN network

Field Name	Type	Description
pSSID	UTF8*	The service set identifier
pBSSID	UTF8*	The basic service set identifier
pFriendlyName	UTF8*	Optional - A name used to identify this network. If not filled, then the name used will be the SSID.
mode	dword	Specifies if the network can be automatically connected if located. • 0x00000000: Manual • 0x00000001: Automatic
hidden	dword	Specifies if the SSID is being actively broadcast • 0x00000000: SSID is broadcast • 0x00000001: SSID is hidden
securityType	WLANSecurityType	The type of security used for this network. See WLANSecurityType
EAPAuthenticationMethod	dword	Optional - The EAP Authentication Method used by the network.
EAP	byte*	Optional - The EAP definition. This could be a proprietary format implementation of the Buffer (to be checked)
EAPSize	dword	Contains the length in bytes of the EAP configuration. If not used should be set to “0”.
encryptionType	WLANEncryptionType	The Encryption Type for WLAN – See WLANEncryptionType definition

keyIndex	dword	Key index - Position of the matching key stored in the Access point/Wireless Router: • 0x00000001: 1 • 0x00000002: 2 • 0x00000003: 3 • 0x00000004: 4
pNetworkKey	UTF8*	Network Key to connect to WLAN Access Point or Wireless router (If not used should be set to "0")

6.2.14 Located_WLANNetwork

Definition Located_WLANNetwork
This prototype defines a structure which describes a WLAN network.

Field Name	Type	Description
Network	WLANNetwork	Please see WLANNetwork
rssi	dword	The signal strength in dBm
known	dword	Identifies if this is a known network • 0x00000000: Unknown • 0x00000001: Known (Known networks are networks SSID or networks identifiers prelisted by the operator or that have already been used/predefined by the user)

6.2.15 ConnectedParameters

Definition ConnectedParameters
This prototype defines a structure which describes an existing network connection (currently applies only to WLAN)

Field Name	Type	Description
Address	IPAddress	The IP Address
SubnetMask	UTF8*	The subnet mask
HttpProxy	UTF8*	The Http proxy.
MACAddress	UTF8*	The MAC address
DefaultGateway	IPAddress	The default Gateway

6.2.16 EAPAuthenticationMethod

Definition EAPAuthenticationMethod
This prototype defines an enumeration of the most commonly EAP authentication methods supported.

EAPAuthenticationMethod	dword	The following EAP Authentication methods are supported (in decimal format accordingly to IANA Extensible Authentication Protocol (EAP) Registry list): • 4: MD5-Challenge • 6: Generic Token Card (GTC) • 13: EAP-TLS • 17: LEAP • 18: EAP-SIM • 21: EAP-TTLS • 23: EAP-AKA • 25: PEAP • 29: EAP MS-CHAP-V2 • 43: EAP-FAST • 47: EAP-PSK • 49: EAP-IKEv2 • 50: EAP-AKA’

6.2.17 SMSRecord

Definition SMSRecord
This prototype defines a structure which describes a SMS record. Note: One SMS Record equals one or more SMS Segments or packages The following words have the same meaning: 'message segment', 'segment', 'SMS segment', 'package' and 'SMS package'

Field Name	Type	Description
msgID	dword	The message ID Note: This ID shall be able to uniquely identify each SMS, including concatenated one. The enabler SHALL combine the concatenated message segments or packages into one SMSRecord associated with only one unique msgID. This parameter has a range 0 to 0xFFFFFFFF, modulus 0x100000000. Value 0 is reserved for special usage here, i.e. sending SMS. Value 0 SHALL NOT be used to identify an SMS record unless it is for sending purpose. See details in CMAPI_SMS_Send ().
msgStatus	dword	A flag to indicate the status of the message • 0x00000000: read • 0x00000001: unread • 0x00000002: sent • 0x00000003: unsent • 0x00000004: draft • 0xFFFFFFFF: unknown Other values, other than the above six types, could be used for SMS stored in terminal device like PC. They are reserved and implementation dependent by the connection manager.
result	dword	The status of message report. • 0x00000000: message delivery successful • 0x00000001: message delivery failed • 0x00000002: message delivery pending, SC is making more transfer attempts • 0xFFFFFFFF: unknown
msgType	dword	The type of message: • 0x00000000: normal message • 0x00000001: message report • 0x00000002: MMS alert • 0x00000003: voice mail • 0xFFFFFFFF: unknown

SMSClass	dword	See [3GPP TS 23.040] for SMS classes definition The class of the SMS message: • 0x00000000: Class 0 Message – not stored • 0x00000001: Class 1 Message - Indicates that this message is to be stored in the local device memory or the SIM/R-UIM/NAA on UICC (depending on memory availability). • 0x00000002: Class 2 Message – used for SIM/R-UIM/NAA on UICC only. This class SHALL only be used if the SMS content was not directly transferred to the "SIM/R-UIM/NAA on UICC" (see ENVELOPE (SMS-PP DOWNLOAD) in [3GPP TS 31.111] or [3GPP2 C.S0035]) • 0x00000003: Class 3 Message – Indicates that this message will be forwarded from the receiving entity to an external device. • 0x00000004: no message class • 0xFFFFFFFF: unknown
totalPack	dword	The total number of packages or segments
currentPack	dword	The number of received packages or segments.
msgLocation	dword	To indicate where the SMS is stored: • 0x00000000: in the SIM/R-UIM/NAA on UICC; • 0x00000001: in the local device; • 0x00000002: in the terminal device, like PC • 0x00000003: not stored. e.g. discarded voice mail messages or display direct messages
time	UTF8*	The time (local time) when the message was received in the inbox or sent in the sentbox/outbox, or saved in the draftbox. The time format should follow: YYYY-MM-DD HH:MM:SS This adheres to ISO 8601

pPhoneNumber	UTF8*	The targeted address(es). Each address shall include its TON (Type Of Number) and NPI (Numbering Plan Identification) parameters (see [3GPP TS 24.008]) coded in binary format (3 binary digits for TON, 4 binary digits for NPI) and separated by a space (i.e.: "<TON> <NPI> <address>"), more than one address could be included, each of them is separated by ',', and "\0\0" indicates end of the addresses, dynamic memory allocation. Informative examples: 1. Numeric representation: "001 0001 8610010\0\0" to represent "+8610010" 2. Alphanumeric representation: "101 0000 Telekom\0\0" to represent "Telekom" 3. Group addresses: "001 0001 8610010, 101 0000 Telekom\0\0" to represent two addresses, i.e. +8610010 and Telekom
pMsgContent	UTF8*	[Optional] The plain text content of the message. The enabler SHALL convert any text message, regardless of the data coding scheme, into UTF8. Note: This field SHALL be available only when msgType is either normal message or message report. The field value SHALL be set to NULL if not available

6.2.18 NAANameType

Definition NAANameType
This prototype defines a structure which describes the NAA name.

Field Name	Type	Description
NAAName	UTF8*	NAA name can be: SIM, R-UIM, USIM_1, USIM_2, ..., USIM_N, CSIM_1, CSIM_2, ..., CSIM_N, ISIM_1, ISIM_2, ..., ISIM_N. If there is no NAA name from the previous list to be associated to one or several AID values available into the UICC (see [ETSI TS 102 221]), then the AID value shall be put in this field.
ApplicationLabel	UTF8*	Application Label (see [ETSI TS 102 221]) corresponding to the NAA or empty if SIM or R-UIM or if there is no Application Label available. It is recommended that the length does not exceed 32 bytes.

6.2.19 PINPUKStatusType

Definition PINPUKStatusType
This prototype defines a structure which describes the information related to the status of the PIN or PUK

Field Name	Type	Description
pNAAName	UTF8*	The name of an active NAA. NAA name can be: SIM, R-UIM, USIM_1, USIM_2, ..., USIM_N, CSIM_1, CSIM_2, ..., CSIM_N, ISIM_1, ISIM_2, ..., ISIM_N. If there is no NAA name from the previous list to be associated to one or several AID values available into the UICC (see [ETSI TS 102 221]), then the AID value shall be put in this field.
Status	byte	The status of the PINs/PUKs. The field is a binary bitmask and MAY indicate multiple values. • Bit 8 to Bit 1 • XXXXXXX0: PIN 1 not verified (PIN 1 lock feature disabled) • XXXXXXX1: PIN 1 verified (PIN 1 lock feature enabled) • XXXXXX0X: PIN 1 disabled • XXXXXX1X: PIN 1 enabled • XXXXX0XX: PIN 1 blocked • XXXXX1XX: PIN 1 unblocked • XXXX0XXX: PUK 1 blocked • XXXX1XXX: PUK 1 unblocked • XXX0XXXX: PIN 2 not verified (PIN 2 lock feature disabled) • XXX1XXXX: PIN 2 verified (PIN 2 lock feature enabled) • XX0XXXXX: PIN 2 disabled • XX1XXXXX: PIN 2 enabled • X0XXXXXX: PIN 2 blocked • X1XXXXXX: PIN 2 unblocked • 0XXXXXXX: PUK 2 blocked • 1XXXXXXX: PUK 2 unblocked
PIN1retry	byte	The number of retry attempts left for the PIN 1 (in decimal format).
PUK1retry	byte	The number of retry attempts left for the PUK 1 (in decimal format).

PIN2retry	byte	The number of retry attempts left for the PIN 2 (in decimal format).
PUK2retry	byte	The number of retry attempts left for the PUK 2 (in decimal format).

6.2.20 CallbackStatus

Definition CallbackStatus
This prototype defines an enumeration of callback status. This is necessary for those callbacks which are initiated with an explicit request.

Status	Type	Description
0X00000000	dword	The function succeeded.
0X00000001	dword	A fatal error has occurred.
0X00000002	dword	The device has entered a power state which does not allow the requested information to be retrieved.
0X00000003	dword	The referenced device is no longer present.
0X00000004	dword	Timeout occurred
0X00000005	dword	Network search timeout

6.2.21 CallbackID

Definition CallbackID
This prototype defines an enumeration of callback ID to register to or unregister from.

Field Name	Type	Description
CallbackID	dword	Callback ID: • 0x00000001: Devices Detection Complete • 0x00000002: Device Changed - Device Addition and Removal • 0x00000003: GetNetworkList Async Complete • 0x00000004: Connect Complete • 0x00000005: Disconnect Complete • 0x00000006: Cancellation of connection Complete • 0x00000007: Session State Change • 0x00000008: Bearer Status Change • 0x00000009: Traffic Channel Dormancy • 0x0000000A: CDMA 2000 Activation State • 0x0000000B: Search WLAN Network Complete • 0x0000000C: Radio Power State Change • 0x0000000D: SetRadioState Async Complete

		• 0x0000000E: Roaming Status Change • 0x0000000F: Signal Strength Change • 0x00000010: GNSS State Change • 0X00000011: SMS Received • 0x00000012: SMS Received with the message • 0x00000013: Not used • 0x00000014: USSD Message Received • 0x00000015: QoS change • 0x00000016: RF Information change • 0x00000017: PIN PUK Status Change • 0x00000018: WLAN Scan complete • 0x00000019: WLAN New network available • 0x0000001A: WLAN Connection Status • 0X0000001B: PUSH message received • 0x0000001C: OMA DM Status Change • 0x0000001D: UICC ToolKit Proactive Command callback • 0x0000001E: UICC Device Terminal Profile callback • 0x0000001F: Verify PIN Needed • 0x00000020: Permitted Bearers Change • 0x00000021: Byte Count • 0x00000022: Connect Secondary PDP Context Complete • 0x00000023: Disconnect Secondary PDP Context Complete • 0x00000024: Cancellation of connection Secondary PDP Context Complete

7. CMAPI-1

7.1 Introduction

The CMAPI-1 interface is mainly a Synchronous Interface with maximum timeout and possibility of cancellation.

However, for long operations (typically more than 7 seconds before the result is available), Asynchronous versions of the API functions are specified in completion of their Synchronous version.

7.2 API Management

7.2.1 CMAPI_API_Open ()

The **CMAPI_API_Open ()** function is used to initialize the OpenCMAPI and also initialize an internal security context. The security request argument is intentionally unspecified. This allows the OpenCMAPI implementations the opportunity to implement innovative and value added security models.

The security request input serves as the credentials which authenticate the caller to the API. It is implementation specific and could consist of a buffer holding a username and password or something more complex such as a certificate. It is the API user responsibility to consult with the service provider in order to understand how to format the security request structure.

Prototype
dword **CMAPI_API_Open** (dword accessLevel, byte* SecurityRequest, dword SecurityRequestSize)

Parameters		
Field Name	**Mode**	**Description**
accessLevel	Input	The access level requested: • 0x00000001 – Connection Manager Application • 0x00000002 – Non Connection Manager Application • 0xF0000000 – 0xFFFFFFFF – Reserved for proprietary access level implementation.
SecurityRequest	Input	The represents a proprietary means of identification and credential presentation to the OpenCMAPI implementation. Each OpenCMAPI vendor is able to customize the type and amount of data to be submitted.
SecurityRequestSize	Input	The size for the buffer in bytes of the security request structure.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0XF0000003	The authentication has been denied. Please seek proper credentials for your access level.
0XF0000004	The security request was malformed. Please consult vendor materials and/or output log.
0XF0000005	The requested access level is not supported.

7.2.2 CMAPI_API_Close ()

The **CMAPI_API_Close ()** function is used to deallocate any internal API structures including the security context.

Prototype
dword **CMAPI_API_Close** ()

Parameters		
Field Name	**Mode**	**Description**

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.

7.2.3 CMAPI_API_GetOpenCMAPIVersion ()

The **CMAPI_API_GetOpenCMAPIVersion ()** function retrieves the version number of the OpenCMAPI used. This call will return the same version number without regard for the device.

Prototype
dword **CMAPI_API_GetOpenCMAPIVersion** (UTF8* pOpenCMAPIVersion, dword* pOpenCMAPIVersionSize)

Parameters		
Field Name	**Mode**	**Description**
pOpenCMAPIVersion	Output	The version number of the OpenCMAPI used. The version number will be formatted in decimal as “x.y.z <vendor specific string> (coded in UTF8 format)”. The x.y.z will indicate the major(x), minor(y), and point (z) release of the API (for example 1.0.0 to identify release 1.0) There will be a single space (“ ”) following the version number if there is a vendor specific string. The vendor specific string is entirely optional and may contain any identification or versioning information the supplier of the API wishes to supply.
pOpenCMAPIVersionSize	Input/Output	The size in byte of pOpenCMAPIVersion buffer

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X30000000	The OpenCMAPIVersion buffer is not large enough
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to

	perform this operation.

7.3 Device Discovery APIs

7.3.1 CMAPI_Discovery_DetectDevices ()

The **CMAPI_Discovery_DetectDevices ()** function causes the OpenCMAPI to actively search for devices. This is a manually triggered operation which requires that the application has registered for **CMAPI_Callback_DetectDevicesComplete**. This operation gives a complete list of devices in the system which are usable from the OpenCMAPI.

Registering for the **CMAPI_Callback_DeviceChanged** is not related to this call. The device changed callback differs as it only gives information when a new device is added or an existing device is removed from the system. Detect devices is used to obtain a list of devices which are currently present in the system.

Prototype
dword **CMAPI_Discovery_DetectDevices** ()

Parameters		
Field Name	**Mode**	**Description**

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.3.2 CMAPI_Discovery_GetDevice ()

The **CMAPI_Discovery_GetDevice ()** function is used to discover information about the devices within the system.

The opaque handle or deviceID is used to eliminate any possible confusion resulting from one device appearing and another disappearing in a short timespan. The deviceID is supplied to the technology specific API calls in order to obtain more detailed information related to the device.

Prototype
dword **CMAPI_Discovery_GetDevice** (dword deviceID, RadioType* pRadio, dword* pDeviceCapability, dword* pConnectionType, dword* pDeviceType, UTF8* pDescription, dword* pDescriptionLength)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The device ID of the device concerned
pRadio	Output	See RadioType definition
pDeviceCapability	Output	The additional capabilities not related to radio type supported by the device: • 0x00000000: No additional capability

		• 0x00000001: GPS • 0x00000002: AGPS in the Control Plane • 0x00000004: AGPS in the User Plane • 0x00000008: Reserved for future use • 0x00000010: Reserved for future use • 0x00000020: Reserved for future use • Any combination of the above
pConnectionType	Output	The type of the device connection: • 0x00000001: USB • 0x00000002: IRDA • 0x00000004: Bluetooth • 0x00000008: Internal Bus • 0x00000010: Serial • 0x00000020: Wi-Fi • 0x00000040: EmulatedEthernet • Any combination of the above
pDeviceType	Output	The type of device this message refers to. • 0x00000001: Embedded modem • 0x00000002: USB modem • 0x00000003: Mobile phone acting as modem • 0x00000004: USB modem with Emulated Ethernet • 0x00000005: Wireless Router
pDescription	Output	The description of the device. Intended to be descriptive and displayed by an application.
pDescriptionLength	Input/Output	On input contains the length of the buffer in bytes of description or if insufficient contains the necessary size.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X3000000E	The description buffer needs to be larger; the description length is set to the minimum number of bytes required.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.3.3 CMAPI_Discovery_OpenDevice ()

The **CMAPI_Discovery_OpenDevice ()** function is used to "open" a device within the system. The device is identified by the UniqueIdentifier obtained as the result of **CMAPI_Callback_DetectDevicesComplete ()** through earlier call to **CMAPI_Discovery_DetectDevices ()**. The function returns an opaque handle or device ID which is used to eliminate any possible confusion resulting from one device appearing and another disappearing in a short timespan. The deviceID is supplied to the technology specific API calls in order to obtain more detailed information related to the device.

Prototype
dword **CMAPI_Discovery_OpenDevice** (UTF8* UniqueIdentifier, dword* pDeviceID)

Parameters		
Field Name	**Mode**	**Description**
UniqueIdentifier	Input	The unique identification of this specific device. The syntax may change from platform to platform, but the unique identifier is guaranteed to be unique to this device on the platform. It MUST not change due to hosting device restart. Example: Windows device GUID.
pDeviceID	Output	An opaque handle which is used to identify and reference this device in other OpenCMAPI calls. The deviceID is valid from the moment the application receives it from CMAPI_Discovery_OpenDevice until it calls CMAPI_Discovery_CloseDevice. During this period it is a reference to this device. After CloseDevice has been called the deviceID has no meaning and should not be used again.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000100	The UniqueIdentifier is referencing a non-existing device
0X00000102	The device is already opened.
0X00000103	Maximum number of device that the API can handle per client is reached (can be 1), close another open device handle.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.3.4 CMAPI_Discovery_CloseDevice ()

The **CMAPI_Discovery_CloseDevice ()** function is used to "close" a device within the system. The device is identified by the deviceID obtained in earlier call to **CMAPI_Discovery_OpenDevice ()**.

Prototype
dword **CMAPI_Discovery_CloseDevice** (dword deviceID)

Parameters		
Field Name	**Mode**	**Description**

deviceID	Input	An opaque handle or deviceID which was obtained in a call to CMAPI_Discovery_OpenDevice. If deviceID is 0, all devices opened by the calling application will be closed. Any outstanding operation will be terminated (e.g. Async operation)

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.4 Cellular Network Management APIs

7.4.1 CMAPI_Network_GetRFInfo ()

The **CMAPI_Network_GetRFInfo ()** function is used to get information about RF (Radio access technology, band class, data rate supported and channel)

Prototype
dword **CMAPI_Network_GetRFInfo** (dword deviceID, RFInfoType* pRFInfoList, dword* pRFInfoListSize, word* pRFInfoListElements)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pRFInfoList	Output	The List of RF Information. See RFInfoType. The RFInfo structures will be laid out at the front of the structure.
pRFInfoListSize	Input/Output	The number of bytes in the RFInfoList buffer.
pRFInfoListElements	Output	The number of elements in the RF Information List

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X30000006	The RFInfoList buffer is not large enough

0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.4.2 CMAPI_Network_GetHomeInformation ()

The **CMAPI_Network_GetHomeInformation ()** function is used to get information about home network of the subscriber for a dedicated System.

Prototype
dword **CMAPI_Network_GetHomeInformation** (dword deviceID, dword systemID, UTF8* pHomeNetworkName, dword* pHomeNetworkNamelength)

Parameters

Field Name	Mode	Description
deviceID	Input	The ID of the device concerned
systemID	Input	The radio system either 3GPP or 3GPP2 to which the function apply when the device is a multi-mode device. • 0x00000000: 3GPP • 0x00000001: 3GPP2
pHomeNetworkName	Output	Numerical value of the MCCMNC of home network (HPLMN) extracted from the system corresponding IMSI followed by the list of the MCCMNC of the EHPLMNs (Equivalent HPLMNs) separated by coma and space ", " (i.e.: HPLMN_MCCMNC, EHPLMN_MCCMNC1, EHPLMN_MCCMNC2, ...). If no EHPLMNs are defined or available the list contains only the HPLMN_MCCMNC
pHomeNetworkNamelength	Input/Output	Buffer length

Return Values

Value	Description
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0x00000003	Buffer not large enough
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000107	System not supported by the device
0X00000130	The device is not in a power state which allows this operation.
0X00000135	No IMSI available.
0X30000030	The buffer is not sufficient to hold the data, the pHomeNetworkNamelength will contain the minimum number of bytes required.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials

	to perform this operation.

7.4.3 CMAPI_Network_GetServingInformation ()

The **CMAPI_Network_GetServingInformation ()** function is used to get information about serving network of the subscriber

Prototype
dword **CMAPI_Network_GetServingInformation** (dword deviceID, NetworkInfoType* pServingNetworkInfo, dword* pServingNetworkInfoListSize, dword* pServingNetworkInfoCount)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pServingNetworkInfo	Output	Network Information (see NetworkInfoType definition) of the serving network(s). In the case of a multimode device, several Serving Network Information outputs are provided.
pServingNetworkInfoListSize	Input/Output	The size of the buffer on input or if insufficient contains the necessary size.
pServingNetworkInfoCount	Output	The total number of elements in the array of ServingNetworkInfo

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000003	Buffer not large enough
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X30000031	The buffer is not sufficient to hold the data, the pServingNetworkInfoListSize will contain the minimum number of bytes required.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.5 Connection Management APIs

7.5.1 CMAPI_NetConnectSrv_MgrCellularProfile ()

The **CMAPI_NetConnectSrv_MgrCellularProfile ()** function is used to manage cellular profiles, including add/delete/update a profile information.

Prototype
dword **CMAPI_NetConnectSrv_MgrCellularProfile** (dword deviceID, dword CellularProfileID, CellularProfileType* CellularProfile, dword Operation)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
CellularProfileID	Input	Cellular Profile ID, the unique identity for a profile. 0xFFFFFFFF is reserved and cannot be used.
CellularProfile	Input	The details information about the profile.
Operation	Input	The operation type to operate the profile, including Add, Delete, Update: • 0x00000001: Add a profile • 0x00000002: Delete a profile • 0x00000003: Update a profile

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000004	Invalid Operation
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00002001	The Cellular profile ID does not exist
0X00002002	The cellular profile ID is not valid
0X00002003	The Cellular profile ID already exists, this only happen when creating a profile with an existing ID
0X00002004	The Cellular profile can not be updated while currently in use (connected)
0X00002101	The user name is not valid
0X00002102	The password is not valid
0X00002104	The APN is not valid
0X00002105	The IP Address is not valid
0X00002106	The primary DNS address is not valid
0X00002107	The secondary DNS address is not valid
0X00002108	The Auth type is not valid
0X00002109	The IPAddrType is not valid
0X0000210A	The profile type is not valid
0X0000210B	The timeout is not valid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated

	credentials to perform this operation.

7.5.2 CMAPI_NetConnectSrv_GetCellularProfile ()

The **CMAPI_NetConnectSrv_GetCellularProfile ()** function is used to get the details of a specific Cellular Profile.

Prototype
dword **CMAPI_NetConnectSrv_GetCellularProfile** (dword deviceID, dword CellularProfileID, CellularProfileType* pCellularProfile, dword* pCellularProfileSize)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
CellularProfileID	Input	The profile ID for the Get operation. 0xFFFFFFFF is reserved and cannot be used.
pCellularProfile	Output	The details for the profile information
pCellularProfileSize	Input/Output	The size of the cellular profile buffer on input or if insufficient contains the necessary size

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00002001	The Cellular profile ID does not exist
0X30000001	The buffer is not sufficient to hold the data, the pCellularProfileSize will contain the minimum number of bytes required.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.5.3 CMAPI_NetConnectSrv_GetCellularProfileList ()

The **CMAPI_NetConnectSrv_GetCellularProfileList ()** function is used to get a list of all Cellular Profile IDs.

Prototype
dword **CMAPI_NetConnectSrv_GetCellularProfileList** (dword deviceID, dword* pCellularProfileIDList, dword* pCellularProfileIDListSize, dword* pCellularProfileIDListCount)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned

pCellularProfileIDList	Output	The buffer to contain the list of profile IDs.
pCellularProfileIDListSize	Input/Output	The size of the buffer on input or if insufficient contains the necessary size.
pCellularProfileIDListCount	Output	Number of entries in the list.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X30000002	The buffer is not sufficient to hold the data, the pCellularProfileIDListSize will contain the minimum number of bytes required.
0X30000028	The structure is not sufficient to hold the data, the CellularProfileIDListSize will contain the minimum number of bytes required.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.5.4 CMAPI_NetConnectSrv_SelectNetwork ()

The **CMAPI_NetConnectSrv_SelectNetwork ()** function is used to select the current network mode and PLMN for a given System.

Prototype
dword **CMAPI_NetConnectSrv_SelectNetwork** (dword deviceID, dword SystemID, RadioType Radio, byte Mode, UTF8* PLMNID)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
systemID	Input	The radio system either 3GPP or 3GPP2 to which the function applies when the device is a multi-mode device. • 0x00000000: 3GPP • 0x00000001: 3GPP2
Radio	Input	Which Radio technology is used = cf. RadioType definition
Mode	Input	The mode to select network mode: • 0x00: automatic network selection • 0x01: manual network selection
PLMNID	Input	The PLMN ID is not used in the case of automatic network selection.

		The PLMNID is coded as a decimal value on the form "MCCMNC".

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000105	The radio references a radio which the device does not support.
0X00000104	The device does not contain hardware which supports this operation.
0X00000107	System not supported by the device
0X00000130	The device is not in a power state which allows this operation.
0X00003101	The requested mode is not valid
0X00003102	The requested PLMNID is not valid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.5.5 CMAPI_NetConnectSrv_GetNetworkList_Sync ()

The **CMAPI_NetConnectSrv_GetNetworkList_Sync ()** will search and compile a list of available Networks. The calling thread will be blocked until the search has completed.

Prototype
dword **CMAPI_NetConnectSrv_GetNetworkList_Sync** (dword deviceID, dword Timeout, NetworkInfoType* pNetworkInfo, dword* pNetworkInfoSize, dword* pNetworkInfoCount)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
Timeout	Input	The maximum time out for the network search (in seconds)
pNetworkInfo	Output	The Network Information (see NetworkInfoType definition) buffer. The NetworkInfo structures will be laid out at the front of the buffer.
pNetworkInfoSize	Input/Output	The size of the network info buffer or if insufficient contains the necessary size.
pNetworkInfoCount	Output	The total number of elements in the array of NetworkInfo

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open

0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X30000003	The size of the network info buffer is insufficient. pNetworkInfoSize contains the minimum number of bytes required.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.5.6 CMAPI_NetConnectSrv_GetNetworkList_Async ()

The **CMAPI_NetConnectSrv_GetNetworkList_Async ()** is used to initiate the search of the Network list. The calling thread returns immediately. The result is reported in callback **CMAPI_Callback_GetNetworkList_Async_Complete ()**.

Prototype
dword **CMAPI_NetConnectSrv_GetNetworkList_Async** (dword deviceID, dword Timeout)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
Timeout	Input	The maximum time for the network search (in seconds).

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.5.7 CMAPI_NetConnectSrv_GetCurrentConnType ()

The **CMAPI_NetConnectSrv_GetCurrentConnType ()** function is used to get the current connection type.

Prototype
dword **NetConnectSrv_GetCurrentConnType** (dword deviceID, dword CellularProfileID, dword* pCurrentConnType)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned

CellularProfileID	Input	Optional - The ID of the Cellular Profile to be used for this function. 0xFFFFFFFF must be used in the optional state.
pCurrentConnType	Output	The connection type: • 0x00000000: DIAL_UP(RAS) • 0x00000001: NDIS • 0x00000002: EmulatedEthernet • 0x00000003: None

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00002001	The Cellular profile ID does not exist
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.5.8 CMAPI_NetConnectSrv_Connect_Async ()

The **CMAPI_NetConnectSrv_Connect_Async ()** function is used to connect to a network. **CMAPI_NetConnectSrv_Connect_Async** is asynchronous; it initiates a connection and then returns immediately. When the connection has finished the Callback **CMAPI_Callback_Connect_Async_Complete** is invoked.

Prototype
dword **CMAPI_NetConnectSrv_Connect_Async** (dword deviceID, dword CellularProfileID, dword ConnType)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
CellularProfileID	Input	The ID of the Cellular Profile to be used for this function. 0xFFFFFFFF is reserved and cannot be used.
ConnType	Input	The connection type: • 0x00000000: DIAL_UP(RAS) • 0x00000001: NDIS • 0x00000002: EmulatedEthernet

Return Values	
Value	**Description**

0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000006	The requested operation cannot currently be completed because another application is currently performing the same operation.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00002001	The Cellular profile ID does not exist
0X00002101	The user name is not valid
0X00002102	The password is not valid
0X00002104	The APN is not valid
0X00002105	The IP Address is not valid
0X00002106	The primary DNS address is not valid
0X00002107	The secondary DNS address is not valid
0X00002108	The Auth type is not valid
0X00002109	The IPAddrType is not valid
0X0000210A	The profile type is not valid
0X0000210B	The timeout is not valid
0X00003001	The requested bearer is not possible
0X00003009	The requested connection type is not valid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.
0XF0000002	The authentication is failed

7.5.9 CMAPI_NetConnectSrv_Disconnect_Async ()

The **CMAPI_NetConnectSrv_Disconnect_Async ()** function is used to disconnect from the network. **CMAPI_NetConnectSrv_Disconnect_Async** is asynchronous; it initiates the disconnect operation and then returns immediately. When the disconnect operation has finished the Callback **CMAPI_Callback_Disconnect_Async_Complete** is invoked.

Prototype
dword **CMAPI_NetConnectSrv_Disconnect_Async** (dword deviceID, dword CellularProfileID)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
CellularProfileID	Input	The ID of the Cellular Profile to be used for this function. 0xFFFFFFFF is reserved and cannot be used.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000006	The requested operation cannot currently be completed because another application is currently performing the same operation.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00002001	The Cellular profile ID does not exist
0X00002101	The user name is not valid
0X00002102	The password is not valid
0X00002104	The APN is not valid
0X00002105	The IP Address is not valid
0X00002106	The primary DNS address is not valid
0X00002107	The secondary DNS address is not valid
0X00002108	The Auth type is not valid
0X00002109	The IPAddrType is not valid
0X0000210A	The profile type is not valid
0X0000210B	The timeout is not valid
0X00003002	There is no connection to disconnect from
0X00003009	The requested connection type is not valid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.5.10 CMAPI_NetConnectSrv_CancelConnect_Async ()

The **CMAPI_NetConnectSrv_CancelConnect_Async ()** function is used to cancel of connect operation (as a result of a call to **CMAPI_NetConnectSrv_Connect_Async**). **CMAPI_NetConnectSrv_CancelConnect_Async** is asynchronous; it initiates the cancelation of an ongoing connect operation and then returns immediately. When the cancellation of the connect operation has finished the Callback **CMAPI_Callback_CancelConnect_Async_Complete** is invoked.

Prototype
dword **CMAPI_NetConnectSrv_CancelConnect_Async** (dword deviceID, dword CellularProfileID)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
CellularProfileID	Input	The ID of the Cellular Profile to be used for this function. 0xFFFFFFFF is reserved and cannot be used.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00002001	The Cellular profile id does not exist
0X00002002	The cellular profile id is not valid
0X00002101	The user name is not valid
0X00002102	The password is not valid
0X00002104	The APN is not valid
0X00002105	The IP Address is not valid
0X00002106	The primary DNS address is not valid
0X00002107	The secondary DNS address is not valid
0X00002108	The Auth type is not valid
0X00002109	The IPAddrType is not valid
0X0000210A	The profile type is not valid
0X0000210B	The timeout is not valid
0X00003004	There is no connecting session for cancellation
0X00003005	The Connection is releasing
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.5.11 CMAPI_NetConnectSrv_SecondaryPDPContext_Connect_Async ()

The **CMAPI_NetConnectSrv_SecondaryPDPContext_Connect_Async ()** function is used to connect to a network. **CMAPI_NetConnectSrv_SecondaryPDPContext_Connect_Async** is asynchronous; it initiates a connection and then returns immediately. When the connection has finished the Callback **CMAPI_Callback_NetConnectSrv_SecondaryPDPContext_Connect_Async_Complete** is invoked.

Prototype
dword **CMAPI_NetConnectSrv_SecondaryPDPContext_Connect_Async** (dword deviceID, dword CellularProfileID, byte SecondaryContextnumber)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
CellularProfileID	Input	The ID of the Cellular Profile to be used for this function. 0xFFFFFFFF is reserved and cannot be used.

SecondaryContext number	Input	Secondary context number from 1 to 16. The API shall check first if a Primary context is activated for this cellular profile The API will check if in the cellular profile the pointer to the Secondary context is set to NULL or not. If not NULL, the function will try to activate the secondary context. The API will also check if this Secondary context is already activated or in progress of activation

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000006	The requested operation cannot currently be completed because another application is currently performing the same operation.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00002001	The Cellular profile ID does not exist
0X00002101	The user name is not valid
0X00002102	The password is not valid
0X00002104	The APN is not valid
0X00002105	The IP Address is not valid
0X00002106	The primary DNS address is not valid
0X00002107	The secondary DNS address is not valid
0X00002108	The Auth type is not valid
0X00002109	The IPAddrType is not valid
0X0000210A	The profile type is not valid
0X0000210B	The timeout is not valid
0X00003001	The requested bearer is not possible
0X00003009	The requested connection type is not valid
0X00003201	No Primary context activated
0X00003202	The secondary context doesn't exist
0X00003203	The secondary context is already activated/created
0X00003204	The secondary context activation is in progress
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.
0XF0000002	The authentication is failed

7.5.12 CMAPI_NetConnectSrv_SecondaryPDPContext_Disconnect_Async ()

The **CMAPI_NetConnectSrv_SecondaryPDPContext_Disconnect_Async ()** function is used to disconnect from the network. **CMAPI_NetConnectSrv_SecondaryPDPContext_Disconnect_Async** is asynchronous; it initiates the disconnect operation and then returns immediately. When the disconnect operation has finished the Callback **CMAPI_Callback_NetConnectSrv_SecondaryPDPContext_Disconnect_Async_Complete** is invoked.

Prototype
dword **CMAPI_NetConnectSrv_SecondaryPDPContext_Disconnect_Async** (dword deviceID, dword CellularProfileID, byte SecondaryContextnumber)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
CellularProfileID	Input	The ID of the Cellular Profile to be used for this function. 0xFFFFFFFF is reserved and cannot be used.
SecondaryContext number	Input	Secondary context number from 1 to 16. The API will check if in the cellular profile the pointer to the Secondary context is set to NULL or not. If not NULL, the function will try to deactivate the secondary context. The API will also check if this Secondary context is already deactivated or in progress of deactivation

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000006	The requested operation cannot currently be completed because another application is currently performing the same operation.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00002001	The Cellular profile ID does not exist
0X00002101	The user name is not valid
0X00002102	The password is not valid
0X00002104	The APN is not valid
0X00002105	The IP Address is not valid
0X00002106	The primary DNS address is not valid
0X00002107	The secondary DNS address is not valid
0X00002108	The Auth type is not valid
0X00002109	The IPAddrType is not valid
0X0000210A	The profile type is not valid

0X0000210B	The timeout is not valid
0X00003002	There is no connection to disconnect from
0X00003009	The requested connection type is not valid
0X00003202	The secondary context doesn't exist
0X00003205	The secondary context is already deactivated
0X00003206	The secondary context deactivation is in progress
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.5.13 CMAPI_NetConnectSrv_SecondaryPDPContext_CancelConnect_Async ()

The **CMAPI_NetConnectSrv_SecondaryPDPContext_CancelConnect_Async ()** function is used to cancel of connect operation (as a result of a call to **CMAPI_NetConnectSrv_SecondaryPDPContext_Connect_Async**). **CMAPI_NetConnectSrv_SecondaryPDPContext_CancelConnect_Async** is asynchronous; it initiates the cancelation of an ongoing connect operation and then returns immediately. When the cancellation of the connect operation has finished, the Callback **CMAPI_Callback_NetConnectSrv_SecondaryPDPContext_CancelConnect_Async_Complete** is invoked.

Prototype
dword **CMAPI_NetConnectSrv_SecondaryPDPContext_CancelConnect_Async** (dword deviceID, dword CellularProfileID, byte SecondaryContextnumber)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
CellularProfileID	Input	The ID of the Cellular Profile to be used for this function. 0xFFFFFFFF is reserved and cannot be used.
SecondaryContext number	Input	Secondary context number from 1 to 16. The API will check if in the cellular profile the pointer to the Secondary context is set to NULL or not. If not NULL, the function will try to activate the secondary context. The API will also check if this Secondary context is already deactivated or in progress of deactivation

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00002001	The Cellular profile ID does not exist

0X00002002	The cellular profile ID is not valid
0X00002101	The user name is not valid
0X00002102	The password is not valid
0X00002104	The APN is not valid
0X00002105	The IP Address is not valid
0X00002106	The primary DNS address is not valid
0X00002107	The secondary DNS address is not valid
0X00002108	The Auth type is not valid
0X00002109	The IPAddrType is not valid
0X0000210A	The profile type is not valid
0X0000210B	The timeout is not valid
0X00003004	There is no connecting session for cancellation
0X00003005	The Connection is releasing
0X00003202	The secondary context doesn't exist
0X00003207	The secondary context is already deactivating
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.6 Network Management APIs

7.6.1 CMAPI_NetCon_GetConnectionStatus ()

The **CMAPI_NetCon_GetConnectionStatus ()** is used to obtain information about the connection status.

Prototype
dword **CMAPI_NetCon_GetConnectionStatus** (dword deviceID, dword CellularProfileID, dword* pConnectionStatus, dword* pTypes, IPAddress* pAddress, dword* pAddressSize, qword* pTxDataRate, qword* pRxDataRate, qword* pTxPackets, qword* pRxPackets, qword* pTxBytes, qword* pRxBytes, dword* pDuration)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
CellularProfileID	Input	Optional - The ID of the Cellular Profile to be used for this function. 0xFFFFFFFF is used in the optional condition.
pConnectionStatus	Output	Connection status values: • 0x00000000: Connected • 0x00000001: Disconnected (it may be possible to distinguish between passive and active disconnection) • 0x00000002: Connecting • 0x00000003: Disconnecting

		• 0x00000004: Scanning • 0x00000010: Unknown state
pTypes	Output	Indication of the radio access technology currently used In the case of a device with multiple radios, there MAY be multiple settings returned. • 0x00000010: GSM service • 0x00000020: GPRS service • 0x00000040: EDGE service • 0x00000080: CDMA service • 0x00000100: QNC service • 0x00000200: 1X-RTT service • 0x00000400: EV-DO service • 0x00000800: EV-DV service • 0x00001000: IOTA service • 0x00002000: IOTA REVA service • 0x00004000: UMTS service • 0x00008000: HSDPA service (Included for legacy purpose, not all operators use HSDPA+) • 0x00010000: HSUPA service • 0x00020000: HSPA Plus service • 0x00040000: PHS service • 0x00080000: FOMA service • 0x00100000: LTE service • 0x10000000: WLAN service
pAddress	Output	IPaddress on interface
pAddressSize	Input/Output	The size of the IPAddress buffer on input. If insufficient, contains the size needed on return.
pTxDataRate	Output	Transmitted Connection Data Rate in Kbit/s
pRxDataRate	Output	Received Connection Data Rate in Kbit/s
pTxPackets	Output	Number of packets transmitted since connection establishment
pRxPackets	Output	Number of packets received since connection establishment
pTxBytes	Output	Number of bytes transmitted since connection establishment
pRxBytes	Output	Number of bytes received since connection establishment
pDuration	Output	Number of seconds elapsed since connection establishment

Return Values	
Value	**Description**

0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00002001	The cellular profile ID does not exist
0X00002002	The Cellular profile ID is not valid
0X30000007	The IPAddress buffer is not sufficient to hold the address. IPAddressSize contains the minimum number of bytes required.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.6.2 CMAPI_NetCon_SetAutoConnectMode ()

The **CMAPI_NetCon_SetAutoConnectMode ()** function is used to set/disable "autoconnect" mode. When the autoconnect functionality is triggered, the default profile for the device will be used to make the connection. The default profile must be set in the **CMAPI_NetCon_SetDefaultProfile** method. If there is need to request the PIN, this will be signalled asynchronously as needed through the callback **CMAPI_Callback_VerifyPIN**. The application should register for the callback before turning on one of the autoconnect modes. If the application does not register and the autoconnect is triggered when a PIN is required, the autoconnect function will not be successful and the application cannot be notified.

Prototype
dword **CMAPI_NetCon_SetAutoConnectMode** (dword deviceID, dword CellularProfileID, dword Mode)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
CellularProfileID	Input	Optional - The ID of the Cellular Profile to be used for this function. 0xFFFFFFFF is used in the optional condition.
Mode	Input	• 0x00000000: Disable autoconnect • 0x00000001: Enable for home network • 0x00000002: Enable for home and roaming network

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00002001	The Cellular profile ID does not exist

0X00002002	The Cellular profile ID is not valid
0X00002005	A default profile has not been set for this device.
0X0000300A	There is currently a connection which prevents this operation. It is necessary to disconnect before the requested operation can be completed.
0X00003101	The requested mode is not valid.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.6.3 CMAPI_NetCon_GetAutoConnectMode ()

The **CMAPI_NetCon_GetAutoConnectMode ()** function is used to return the current "autoconnect" mode.

Prototype
dword **CMAPI_NetCon_GetAutoConnectMode** (dword deviceID, dword CellularProfileID, dword* pMode)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
CellularProfileID	Input	Optional - The ID of the Cellular Profile to be used for this function. 0xFFFFFFFF is used in the optional condition.
pMode	Output	• 0x00000000: Disable autoconnect • 0x00000001: Enable for home network • 0x00000002: Enable for home and roaming network

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00002001	The Cellular profile ID does not exist
0X00002002	The Cellular profile ID is not valid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.6.4 CMAPI_NetCon_SetDefaultProfile ()

The **CMAPI_NetCon_SetDefaultProfile ()** function is used to identify the profile that shall be used when the device is in auto connect mode (See **CMAPI_NetCon_SetAutoConnectMode**).

Prototype
dword **CMAPI_NetCon_SetDefaultProfile** (dword deviceID, dword CellularProfileIDdefault)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
CellularProfileIDdefault	Input	The cellular profile ID per default (reference CellularProfileID). 0xFFFFFFFF is reserved and cannot be used.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00002001	The Cellular profile ID does not exist
0X00002002	The Cellular profile ID is not valid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.6.5 CMAPI_NetCon_SetPermittedBearers ()

The **CMAPI_NetCon_SetPermittedBearers ()** function is used to restrict the permitted mobile bearer when connecting to the selected network.

Prototype
dword **CMAPI_NetCon_SetPermittedBearers** (dword deviceID, dword Bearers)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
Bearers	Input	Bearer (s) selected: • 0x00000001: GSM • 0x00000002: WCDMA/UMTS • 0x00000004: CDMA

		• 0x00000008: EVDO • 0x00000010: TD_SCDMA • 0x00000020: LTE Automatic will be realized by selecting multiple bearers in the bitmap

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000010	The OpenCMAPI implementation cannot perform this operation since there is currently a connection which prevents the request. NOTE: The OpenCMAPI implementation may be able to apply the change in some conditions and may return success instead of this return code in some connected conditions.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000120	This configuration is not supported
0X00000121	The device does not offer this capability
0X00000130	The device is not in a power state which allows this operation.
0X00003103	The requested bearer or combination of bearers is not valid.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.6.6 CMAPI_NetCon_GetPermittedBearers ()

The **CMAPI_NetCon_GetPermittedBearers ()** function is used to get the current permitted bearers.

Prototype
dword **CMAPI_NetCon_GetPermittedBearers** (dword deviceID, dword* pBearers)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pBearers	Output	Bearer (s) selected: • 0x00000001: GSM • 0x00000002: WCDMA/UMTS • 0x00000004: CDMA • 0x00000008: EVDO • 0x00000010: TD_SCDMA • 0x00000020: LTE Automatic will be realized by selecting multiple bearers in the bitmap

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.6.7 CMAPI_NetCon_SetNoDataProfile ()

The **CMAPI_NetCon_SetNoDataProfile ()** function is used to set up (enabled or disabled) the nodataprofile. The nodataprofile is used, for example, to simulate in LTE the equivalent of Attachment in 3G as in LTE, there is no similar behaviour - always connected.

Prototype
dword **CMAPI_NetCon_SetNoDataProfile** (dword deviceID, dword CellularProfileID, dword State)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
CellularProfileID	Input	Cellular Profile ID, the unique identity for a profile. 0xFFFFFFFF is reserved and cannot be used.
State	Input	To indicate if the Nodataprofile needs to be enabled or not: • 0x00000000: disabled • 0x00000001: enabled

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00002001	The Cellular profile ID does not exist
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.6.8 CMAPI_NetCon_GetNoDataProfile ()

The **CMAPI_NetCon_GetNoDataProfile ()** function is used to return the current state of the nodata profile (enabled or disabled).

Prototype
dword **CMAPI_NetCon_GetNoDataProfile** (dword deviceID, dword* pCellularProfileID, dword* pState)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pCellularProfileID	Output	Cellular Profile ID, the unique identity for a profile. 0xFFFFFFFF is reserved and cannot be used.
pState	Output	To indicate if the Nodataprofile is enabled or not: • 0x00000000: disabled • 0x00000001: enabled

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7 CDMA2000 APIs

7.7.1 CMAPI_CDMA2000_SetACCOLC ()

The **CMAPI_CDMA2000_SetACCOLC ()** function is used to set the Access Overload Class (ACCOLC) for CDMA2000 devices.

Prototype
dword **CMAPI_CDMA2000_SetACCOLC** (dword deviceID, UTF8* SPC, byte Accolc)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
SPC	Input	The Service Programming Code (SPC).
Accolc	Input	New value of Access Overload Class parameter (range 0 to 15).

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00004003	The SPC is invalid
0X0000400C	The ACCOLC is invalid.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.2 CMAPI_CDMA2000_GetACCOLC ()

The **CMAPI_CDMA2000_GetACCOLC ()** function is used to retrieve the current value of the Access Overload Class (ACCOLC) for CDMA2000 devices.

Prototype
dword **CMAPI_CDMA2000_GetACCOLC** (dword deviceID, byte* pAccolc)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pAccolc	Output	Pointer to current value of Access Overload Class parameter (range 0 to 15).

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.3 CMAPI_CDMA2000_SetCDMANetworkParameters ()

The **CMAPI_CDMA2000_SetCDMANetworkParameters ()** function is used to set the values of certain CDMA2000-specific network parameters.

Prototype
dword **CMAPI_CDMA2000_SetCDMANetworkParameters** (dword deviceID, UTF8* SPC, dword ForceRev0, dword CustomSCP, dword Protocol, dword Broadcast, dword Application, dword Roaming)

Parameters

Field Name	Mode	Description
deviceID	Input	The ID of the device concerned
SPC	Input	The Service Programming Code (SPC).
ForceRev0	Input	(Optional) Force CDMA 1x-EV-DO Rev. 0 mode
CustomSCP	Input	(Optional) Use a custom config for CDMA 1x-EV-DO SCP
Protocol	Input	(Optional) Protocol mask for custom SCP config
Broadcast	Input	(Optional) Custom mask for broadcast Session Configuration Protocol (SCP) configuration: • 0x00000001: Generic broadcast enabled • All other values (except 0xFFFFFFFF) reserved for future use
Application	Input	(Optional) Application mask for custom SCP configuration: • 0x00000001: SN multiflow packet application • 0x00000002: Enhanced SN multiflow packet application • All other values (except 0xFFFFFFFF) reserved for future use
Roaming	Input	(Optional) Roaming preference: • 0x00000000: Automatic • 0x00000001: Home only • 0x00000002: Affiliated only (restrict roaming to a network having a roaming agreement (affiliation) with the Home operator) • 0x00000003: Home and Affiliated

Return Values

Value	Description
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00004003	The SPC is invalid
0X0000400D	The requested ForceRev0 is invalid
0X0000400E	The CustomSCP is invalid
0X0000400F	The protocol is invalid

0X00004010	The broadcast is invalid
0X00004011	The application is invalid
0X00004012	The roaming is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.4 CMAPI_CDMA2000_GetCDMANetworkParameters ()

The **CMAPI_CDMA2000_GetCDMANetworkParameters ()** function is used to retrieve the values of certain CDMA2000-specific network parameters.

Prototype
dword **CMAPI_CDMA2000_GetCDMANetworkParameters** (dword deviceID, byte* pSCI, byte* pSCM, byte* pRegHomeSID, byte* pRegForeignSID, byte* pRegForeignNID, dword* pBroadcast, dword* pApplication, dword* pRoaming)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pSCI	Output	Slot cycle index (0xFF if unknown)
pSCM	Output	Station class mark (0xFF if unknown)
pRegHomeSID	Output	Registration on home system: • 0x00: Disabled • 0x01: Enabled • 0xFF: Unknown
pRegForeignSID	Output	Registration on foreign system: • 0x00: Disabled • 0x01: Enabled • 0xFF: Unknown
pRegForeignNID	Output	Registration on foreign network: • 0x00: Disabled • 0x01: Enabled • 0xFF: Unknown
pBroadcast	Output	Custom mask for broadcast Session Configuration Protocol (SCP) configuration: • 0x00000001: Generic broadcast enabled • 0xFFFFFFFF: Unknown • All other values reserved for future use
pApplication	Output	Application mask for custom SCP configuration: • 0x00000001: SN multiflow packet application

		• 0x00000002: Enhanced SN multiflow packet application • 0xFFFFFFFF: Unknown • All other values reserved for future use
pRoaming	Output	Roaming preference: • 0x00000000: Automatic • 0x00000001: Home only • 0x00000002: Affiliated only • 0x00000003: Home and Affiliated • 0xFFFFFFFF: Unknown

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.5 CMAPI_CDMA2000_GetANAAAAAuthenticationStatus ()

The **CMAPI_CDMA2000_GetANAAAAAuthenticationStatus ()** function is used to retrieve the value of the most recent ANA AAA authentication attempt status for CDMA2000 devices.

Prototype
dword **CMAPI_CDMA2000_GetANAAAAAuthenticationStatus** (dword deviceID, dword* pStatus)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pStatus	Output	Outcome of the most recent ANA AAA authentication attempt: • 0x00000000: Failure • 0x00000001: Success • 0x00000002: Not attempted

Return Values	
Value	**Description**
0X00000000	The function succeeded.

0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.6 CMAPI_CDMA2000_GetPRLVersion ()

The **CMAPI_CDMA2000_GetPRLVersion ()** function is used to retrieve the value of the Preferred Roaming List (PRL) version in use for CDMA2000 devices.

Prototype
dword **CMAPI_CDMA2000_GetPRLVersion** (dword deviceID, word* pPRLVersion)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pPRLVersion	Output	PRL version number

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.7 CMAPI_CDMA2000_GetERIFile ()

The **CMAPI_CDMA2000_GetERIFile ()** function is used to retrieve the contents of the Enhanced Roaming Indicator (ERI) file in use for CDMA2000 devices.

Prototype
dword **CMAPI_CDMA2000_GetERIFile** (dword deviceID, byte* pFile, dword* pFileSize)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pFile	Output	Pointer to memory area that will contain contents of the ERI file when this function returns.
pFileSize	Input/Output	On input, contains the maximum number of bytes that can be stored in the memory area pointed to by pFile; on output, contains the number of bytes actually written to the memory area by GetERIFile or if insufficient contains the necessary size.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X30000008	The pFile buffer was insufficient; pFileSize contains the minimum number of bytes required.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.8 CMAPI_CDMA2000_ActivateAutomatic ()

The **CMAPI_CDMA2000_ActivateAutomatic ()** function commands the device to perform automatic activation using a specified activation code.

Prototype
dword **CMAPI_CDMA2000_ActivateAutomatic** (dword deviceID, UTF8* ActivationCode)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
ActivationCode	Input	The activation code (maximum length is 12 characters).

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00004004	The requested activation code is invalid.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.9 CMAPI_CDMA2000_ActivateManual ()

The **CMAPI_CDMA2000_ActivateManual ()** function commands the device to perform manual activation using the specified parameters.

Prototype
dword **CMAPI_CDMA2000_ActivateManual** (dword deviceID, UTF8* SPC, word SID, UTF8* MDN, UTF8* MIN, dword PRLSize, UTF8* PRL, UTF8* MNHA, UTF8* MNAAA)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
SPC	Input	The 6-digit Service Programming Code (SPC)
SID	Input	System identification number (SID)
MDN	Input	Mobile Directory Number (MDN) value
MIN	Input	Mobile Identity Number (MIN) value
PRL	Input	(Optional) PRL file contents
PRLSize	Input	(Optional) Size in bytes of the Preferred Roaming List (PRL)
MNHA	Input	(Optional) MN-HA value
MNAAA	Input	(Optional) MN-AAA value

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.

0X00004003	The SPC is invalid
0X00004013	The SID is invalid
0X00004014	The MDN is invalid
0X00004015	The MIN is invalid
0X00004016	The PRL is invalid
0X00004017	The MNHA is invalid
0X00004018	The MNAAA is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.10 CMAPI_CDMA2000_ValidateSPC ()

The **CMAPI_CDMA2000_ValidateSPC ()** function commands the device to validate a Service Programming Code (SPC) [3GPP2 C.S0016].

Prototype
dword **CMAPI_CDMA2000_ValidateSPC** (dword deviceID, UTF8* SPC)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
SPC	Input	The SPC (six-digit value)

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00004003	The SPC is invalid.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.11 CMAPI_OMADM_StartSession ()

The **CMAPI_OMADM_StartSession ()** function starts an OMA DM session to configure the values of various CDMA2000 network information as specified by the session type in its input parameter.

Prototype
dword **CMAPI_OMADM_StartSession** (dword deviceID, dword SessionType, dword* pSessionIdentifier)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
SessionType	Input	Type of session to be started: • 0x00000000: Client-initiated device configuration • 0x00000001: Client-initiated PRL update • 0x00000002: Client-initiated hands-free activation • 0x00000006: (optional) Client-initiated Firmware Update
pSessionIdentifier	Output	Identifies the session and which can be referenced when required, such as tracking active sessions, cancelling the session, etc.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00004019	The session type is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.12 CMAPI_OMADM_CancelSession ()

The **CMAPI_OMADM_CancelSession ()** cancels an ongoing OMA DM session.

Prototype
dword **CMAPI_OMADM_CancelSession** (dword deviceID, dword sessionIdentifier)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
sessionIdentifier	Input	(Optional) The session identifier which was returned when the session was started.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.

0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00004001	Unrecognized session identifier.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.13 CMAPI_OMADM_GetSessionInfo ()

The **CMAPI_OMADM_GetSessionInfo ()** function returns information about the currently active OMA DM session (or the most recent session if none is active).

Prototype
dword **CMAPI_OMADM_GetSessionInfo** (dword deviceID, dword* pSessionType, dword* pSessionState, dword* pFailureReason, byte* pRetryCount, word* pSessionPause, word* pTimeRemaining)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pSessionType	Output	Type of session: • 0x00000000: Client-initiated device configuration • 0x00000001: Client-initiated PRL update • 0x00000002: Client-initiated hands-free activation • 0x00000003: Device-initiated hands-free activation • 0x00000004: Network-initiated PRL update • 0x00000005: Network-initiated device configuration • 0x00000006: (optional) Client-initiated firmware update • 0x00000007: (optional) Network-initiated firmware update
pSessionState	Output	State of the session: • 0x00000000: Complete, information was updated • 0x00000001: Complete, update information unavailable • 0x00000002: Complete, no new update available • 0x00000003: Failed • 0x00000004: Retrying • 0x00000005: Connecting • 0x00000006: Connected • 0x00000007: Authenticated • 0x00000008: Mobile Directory Number (MDN) downloaded • 0x00000009: Mobile Station Identifier (MSID) downloaded

		• 0x0000000A: PRL downloaded • 0x0000000B: Mobile IP profile downloaded
pFailureReason	Output	Session failure reason: • 0x00000000: Unknown • 0x00000001: Network is unavailable • 0x00000002: Server is unavailable • 0x00000003: Authentication failed • 0x00000004: Maximum number of retries exceeded • 0x00000005: Session is canceled
pRetryCount	Output	Session retry count
pSessionPause	Output	Time (in seconds) to pause between retries
pTimeRemaining	Output	Time (in seconds) remaining until next retry (when session state is Retrying)

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.14 CMAPI_OMADM_GetPendingNIA ()

The **CMAPI_OMADM_GetPendingNIA ()** function returns information about a Network-Initiated Alert (NIA) that is commanding the device to establish a DM session with a DM server to perform the requested configuration operation.

Prototype
dword **CMAPI_OMADM_GetPendingNIA** (dword deviceID, dword* pSessionType, word* pSessionID)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pSessionType	Output	Type of session to be started: • 0x00000004: Network-initiated PRL update • 0x00000005: Network-initiated device configuration • 0x00000007: (optional) Network-initiated firmware update

pSessionID	Output	Session ID for the NIA request

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.15 CMAPI_OMADM_SendSelection ()

The **CMAPI_OMADM_SendSelection ()** returns the response from the device to a Network-Initiated Alert (NIA) that is commanding the device to establish a DM session. The device/user can either reject or accept the session request from the network.

Prototype
dword **CMAPI_OMADM_SendSelection** (dword deviceID, dword selection, dword sessionID, dword defer)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
selection	Input	Response selected to the NIA: • 0x00000000: Reject • 0x00000001: Accept • 0x00000002: Defer
sessionID	Input	Session ID from the NIA request
defer	Input	Specifies that the server is able to defer. • 0x00000000: Defer allowed • 0x00000001: Defer is not allowed

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.

0X00000130	The device is not in a power state which allows this operation.
0X0000401E	The selection is invalid
0X0000401F	The session id is invalid
0X00004020	The defer is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.16 CMAPI_OMADM_GetFeatureSettings ()

The **CMAPI_OMADM_GetFeatureSettings ()** function returns information about the settings of OMA DM features, indicating for each one whether OMA DM can be currently used for the specified configuration operation.

Prototype
dword **CMAPI_OMADM_GetFeatureSettings** (dword deviceID, dword* pProvisioning, dword* pPRLUpdate, dword* pFirmwareUpdate)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pProvisioning	Output	Setting of device provisioning service update feature: • 0x00000000: Disabled • 0x00000001: Enabled
pPRLUpdate	Output	Setting of PRL service update feature: • 0x00000000: Disabled • 0x00000001: Enabled
pFirmwareUpdate	Output	Setting of Firmware update feature: • 0x00000000: Disabled • 0x00000001: Enabled • 0xFFFFFFFF: Not supported

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.17 CMAPI_OMADM_SetProvisioningFeature ()

The **CMAPI_OMADM_SetProvisioningFeature ()** function is used to enable and disable the OMA DM device service provisioning update feature.

Prototype
dword **CMAPI_OMADM_SetProvisioningFeature** (dword deviceID, dword provFeatureState)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
provFeatureState	Input	State of device provisioning service update: • 0x00000000: Disable • 0x00000001: Enable

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00004021	The feature state is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.18 CMAPI_OMADM_SetPRLUpdateFeature ()

The **CMAPI_OMADM_SetPRLUpdateFeature ()** function is used to enable and disable the OMA DM PRL update feature.

Prototype
dword **CMAPI_OMADM_SetPRLUpdateFeature** (dword deviceID, dword PRLUpdateFeatureState)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
PRLUpdateFeatureState	Input	State of PRL update feature: • 0x00000000: Disable • 0x00000001: Enable

Return Values

Value	Description
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00004022	The update feature state is invalid.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.19 CMAPI_OMADM_SetFirmwareUpdateFeature () (Optional)

The **CMAPI_OMADM_SetFirmwareUpdateFeature ()** function is used to enable and disable the OMA DM Firmware update feature.

Prototype
dword **CMAPI_OMADM_SetFirmwareUpdateFeature** (dword deviceID, dword firmwareUpdateFeatureState)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
firmwareUpdateFeatureState	Input	State of Firmware update feature: • 0x00000000: Disable • 0x00000001: Enable

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000007	This optional function is not supported by this implementation
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00004023	The firmware update feature state is invalide
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.20 CMAPI_OMADM_ResetToFactoryDefaults ()

The **CMAPI_OMADM_ResetToFactoryDefaults ()** function is used to reset the device to factory default.

Prototype
dword **CMAPI_OMADM_ResetToFactoryDefaults** (dword deviceID, dword SPCCode, dword Reason)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
SPCCode	Input	(Optional) Valid SPC
Reason	Input	• 0x00000001: PRI Update • 0x00000002: RTN Reset (SPC required)

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00004003	The SPC is invalid.
0X00004024	The reason is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.21 CMAPI_OMADM_InitiateOTASP ()

The **CMAPI_OMADM_InitiateOTASP ()** function is used for activating the device using OTA activation. This function allows configuring parameters such as MDN, MIN, Home SID, MN-HA and AAA key. Existing PRL may also be replaced with a new PRL.

Prototype
dword **CMAPI_OMADM_InitiateOTASP** (dword deviceID, UTF8* ActivationCode)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
ActivationCode	Input	Valid Activation Code (SPC code)

Return Values	
Value	**Description**
0X00000000	The function succeeded.

0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000110	The device cannot be activated while connected.
0X00000130	The device is not in a power state which allows this operation.
0X00004004	The requested activation code is invalid.
0X00004005	Activation failed (other than invalid activation code).
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.22 CMAPI_OMADM_SetPRL ()

The **CMAPI_OMADM_SetPRL ()** function is used to update PRL/PLMN by uploading a PRL file.

Prototype
dword **CMAPI_OMADM_SetPRL** (dword deviceID, UTF8* PRLFilepath)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
PRLFilepath	Input	Valid PRL file path.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00004007	File does not exist at the given path.
0X00004008	An invalid PRL file is entered.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.23 CMAPI_MobileIP_SetState ()

The **CMAPI_MobileIP_SetState ()** function is used to set the current Mobile IP state of the device.

Prototype
dword **CMAPI_MobileIP_SetState** (dword deviceID, dword Mode)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
Mode	Input	The new setting of the device's Mobile IP mode: • 0x00000000: Mobile IP off (simple IP only) • 0x00000001: Mobile IP preferred • 0x00000002: Mobile IP only

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00004025	The mode is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.24 CMAPI_MobileIP_GetState ()

The **CMAPI_MobileIP_GetState ()** function is used to retrieve the current Mobile IP state of the device.

Prototype
dword **CMAPI_MobileIP_GetState** (dword deviceID, dword* pMode)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pMode	Output	Pointer to the current setting of the device's Mobile IP mode: • 0x00000000: Mobile IP off (simple IP only) • 0x00000001: Mobile IP preferred • 0x00000002: Mobile IP only

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.

0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.25 CMAPI_MobileIP_SetActiveProfile ()

The **CMAPI_MobileIP_SetActiveProfile ()** function is used to set the index of the Mobile IP profile that the device will use. There can be several Mobile IP profiles configured on the device, each of which is identified by a unique index.

Prototype
dword **CMAPI_MobileIP_SetActiveProfile** (dword deviceID, UTF8* SPC, byte index)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
SPC	Input	The Service Programming Code (SPC).
index	Input	Index of the mobile IP profile that will be made the active one.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00004003	The SPC is invalid
0X00004006	The index is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.26 CMAPI_MobileIP_GetActiveProfile ()

The **CMAPI_MobileIP_GetActiveProfile ()** function is used to retrieve the index of the Mobile IP profile that the device is currently using.

Prototype
dword **CMAPI_MobileIP_GetActiveProfile** (dword deviceID, byte* pIndex)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pIndex	Output	Pointer to the index of the currently active mobile IP profile.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.27 CMAPI_MobileIP_SetProfile ()

The **CMAPI_MobileIP_SetProfile ()** function is used to configure the contents of a Mobile IP profile on the device. The function takes as arguments the index of the Mobile IP profile that will be modified and the profile values that will be set by the function.

Prototype
dword **CMAPI_MobileIP_SetProfile** (dword deviceID, UTF8* SPC, byte index, byte Enabled, IPAddress* Address, IPAddress* PriHA, IPAddress* SecHA, byte RevTunn, UTF8* NAI, dword HASPI, dword AAASPI, UTF8* MNHA, UTF8* MNAAA)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
SPC	Input	The Service Programming Code (SPC).
index	Input	Index of the mobile IP profile that is being set with this function.
Enabled	Input	(Optional) Enable profile: • 0x00: No (disable), • any other value except 0xFF: Yes (enable)

Address	Input	(Optional) Home IP address
PriHA	Input	(Optional) Primary Home Agent IP address
SecHA	Input	(Optional) Secondary Home Agent IP address
RevTunn	Input	(Optional) Reverse tunneling mode: • 0x00: No (Disable), • any other value except 0xFF: Enable
NAI	Input	(Optional) Network Access Identifier
HASPI	Input	(Optional) Home Agent Security Parameter Index
AAASPI	Input	(Optional) AAA server Security Parameter Index
MNHA	Input	(Optional) MN-HA key
MNAAA	Input	(Optional) AAA key

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00004003	The SPC is invalid
0X00004006	The index is invalid
0X00004017	The MNHA is not valid
0X00004018	The MNAAA is not valid
0X00004026	The enabled value is not valid
0X00004027	The RevTunn value is not valid
0X00004028	The NAI is not valid
0X00004029	The HASPI is not valid
0X0000402A	The AAASPI is not valid
0X0000402B	The Address parameter was not formatted properly.
0X0000402C	The Primary Home Agent parameter was not formatted properly.
0X0000402D	The Secondary Home Agent parameter was not formatted properly.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.28 CMAPI_MobileIP_GetProfile ()

The **CMAPI_MobileIP_GetProfile ()** function is used to retrieve the contents of a Mobile IP profile on the device. The function takes as arguments the index of the Mobile IP profile that will be retrieved and the profile values that will be returned by the function.

Prototype
dword **CMAPI_MobileIP_GetProfile** (dword deviceID, byte index, byte* pEnabled, IPAddress* pAddress, dword* pAddressSize, IPAddress* pPriHA, dword* pPriHASize, IPAddress* pSecHA, dword* pSecHASize, byte* pRevTunn, UTF8* pNAI, dword* pNAISize, dword* pHASPI, dword* pAAASPI, dword* pHAState, dword* pAAAState)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
index	Input	Index of the mobile IP profile that is being set with this function.
pEnabled	Output	Profile status: • 0x00: Disabled; • 0x01: Enabled; • 0xFF: Unknown
pAddress	Output	Home IP address
pAddressSize	Input/Output	The size of the address buffer
pPriHA	Output	Primary Home Agent IP address
pPriHASize	Input/Output	The size of the primary home agent buffer
pSecHA	Output	Secondary Home Agent IP address
pSecHASize	Input/Output	The size of the secondary home agent buffer
pRevTunn	Output	Reverse tunneling status: • 0x00: Disabled • 0x01: Enabled • 0xFF: Unknown
pNAI	Output	Network Access Identifier
pNAISize	Input/Output	Number of bytes in the NAI buffer or if insufficient contains the necessary size
pHASPI	Output	Home Agent Security Parameter Index (0xFFFFFFFF: Unknown)
pAAASPI	Output	AAA server Security Parameter Index (0xFFFFFFFF: Unknown)
pHAState	Output	Home Agent Key state: • 0x00000000: Unset • 0x00000001: Set, default value • 0x00000002: Set, non-default value • 0xFFFFFFFF: Unknown

pAAAState	Output	AAA Key state: • 0x00000000: Unset • 0x00000001: Set, default value • 0x00000002: Set, non-default value • 0xFFFFFFFF: Unknown

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X3000000A	The NAI buffer is insufficient. pNAISize contains the minimum number of bytes required.
0X3000000B	The address buffer is insufficient. The size parameter contains the minimum required byte size.
0X3000000C	The primary ha address buffer is insufficient. The size parameter contains the minimum required byte size.
0X3000000D	The secondary ha address buffer is insufficient. The size parameter contains the minimum required byte size.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.29 CMAPI_MobileIP_SetParameters ()

The **CMAPI_MobileIP_SetParameters ()** function is used to set various parameters that configure the behavior of the device's Mobile IP client.

Prototype
dword **CMAPI_MobileIP_SetParameters** (dword deviceID, UTF8* SPC, dword Mode, byte RetryLimit, byte RetryInterval, byte ReRegPeriod, byte ReRegTraffic, byte HAAuthenticator, byte HA2002bis)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
SPC	Input	Service Programming Code (SPC)
Mode	Input	(Optional) Mobile IP mode: • 0x00000000: Mobile IP off (simple IP only) • 0x00000001: Mobile IP preferred • 0x00000002: Mobile IP only
RetryLimit	Input	(Optional) Mobile IP registration attempt retry limit
RetryInterval	Input	(Optional) Mobile IP registration attempt retry interval (i.e. time between registration attempts) in minutes
ReRegPeriod	Input	(Optional) Mobile IP re-registration period (time after which current registration expires) in minutes
ReRegTraffic	Input	(Optional) Determines whether to re-register only if there has been data traffic since last registration: • 0x00: Disable • any other value except 0xFF: Enable
HAAuthenticator	Input	(Optional) State of MH-HA authenticator calculator: • 0x00: Disable • any other value except 0xFF: Enable
HA2002bis	Input	(Optional) Determines whether to use RFC2002bis authentication instead of RFC2002: • 0x00: RFC2002 • any other value except 0xFF: RFC2002bis

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.

0X00000130	The device is not in a power state which allows this operation.
0X00004003	The SPC is invalid
0X00004025	The mode is invalid
0X0000402E	The retry limit is invalid
0X0000402F	The retry interval is invalid
0X00004030	The Reregperiod is invalid
0X00004031	The Reregtraffic is invalid
0X00004032	The HAAuthenticator is invalid
0X00004033	The HA2002bis is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.30 CMAPI_MobileIP_GetParameters ()

The **CMAPI_MobileIP_GetParameters ()** function is used to retrieve the current values of the parameters that configure the behavior of the device's Mobile IP client.

Prototype
dword **CMAPI_MobileIP_GetParameters** (dword deviceID, dword* pMode, byte* pRetryLimit, byte* pRetryInterval, byte* pReRegPeriod, byte* pReRegTraffic, byte* pHAAuthenticator, byte* pHA2002bis)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pMode	Output	Mobile IP mode: • 0x00000000: Mobile IP off (simple IP only) • 0x00000001: Mobile IP preferred • 0x00000002: Mobile IP only • 0xFFFFFFFF: Unknown
pRetryLimit	Output	Mobile IP registration attempt retry limit (0xFF if unknown)
pRetryInterval	Output	Mobile IP registration attempt retry interval (i.e. time between registration attempts) in minutes (0xFF if unknown)
pReRegPeriod	Output	Mobile IP re-registration period (time after which current registration expires) in minutes (0xFF if unknown)
pReRegTraffic	Output	Determines whether to re-register only if there has been data traffic since last registration: • 0x00: Disabled • 0x01: Enabled • 0xFF: Unknown
pHAAuthenticator	Output	State of MH-HA authenticator calculator:

		• 0x00: Disabled • 0x01: Enabled • 0xFF: Unknown
pHA2002bis	Output	(Optional) Determines whether to use RFC2002bis authentication instead of RFC2002: • 0x00: Disabled • 0x01: Enabled • 0xFF: Unknown

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.7.31 CMAPI_MobileIP_GetLastError ()

The **CMAPI_MobileIP_GetLastError ()** function is used to retrieve the last Mobile IP error that occurred (refer to RFC3344 for a list of error codes).

Prototype
dword **CMAPI_MobileIP_GetLastError** (dword deviceID, dword* pError)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pError	Output	Pointer to the most recent Mobile IP error code: • 0x00000000: Success • Any other value except 0xFFFFFFFF: Error code as defined in [RFC3344]

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.

0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.8 Device Service APIs

7.8.1 CMAPI_DevSrv_GetManufacturerName ()

The **CMAPI_DevSrv_GetManufacturerName ()** function retrieves the name of the manufacturer of the device.

Prototype
dword **CMAPI_DevSrv_GetManufacturerName** (dword deviceID, UTF8* pManufacturerName, dword* pManufacturerNameSize)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pManufacturerName	Output	The name of the device manufacturer
pManufacturerNameSize	Input/Output	The size in byte of pManufacturerName buffer

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000130	The device is not in a power state which allows this operation.
0X30000010	The manufacturer name buffer is not large enough.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.8.2 CMAPI_DevSrv_GetManufacturerModel ()

The **CMAPI_DevSrv_GetManufacturerModel ()** function retrieves the product model ID of the device.

Prototype
dword **CMAPI_DevSrv_GetManufacturerModel** (dword deviceID, UTF8* pModel, dword* pModelSize)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pModel	Output	The product model ID of the device
pModelSize	Input/Output	The size in byte of pModel buffer

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000130	The device is not in a power state which allows this operation.
0X30000011	The Model buffer is not large enough.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.8.3 CMAPI_DevSrv_GetDeviceName ()

The **CMAPI_DevSrv_GetDeviceName ()** function retrieves the commercial name of the Device.

Prototype
dword **CMAPI_DevSrv_GetDeviceName** (dword deviceID, UTF8* pDeviceName, dword* pDeviceNameSize)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pDeviceName	Output	The commercial name of the Device
pDeviceNameSize	Input/Output	The size in byte of pDeviceName buffer

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.

0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation
0X00000130	The device is not in a power state which allows this operation.
0X30000012	The device name buffer is not large enough.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.8.4 CMAPI_DevSrv_GetHardwareVersion ()

The **CMAPI_DevSrv_GetHardwareVersion ()** function retrieves the hardware version of the Device.

Prototype
dword **CMAPI_DevSrv_GetHardwareVersion** (dword deviceID, UTF8* pHardwareVersion, dword* pHardwareVersionSize)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pHardwareVersion	Output	The hardware version of the Device
pHardwareVersionSize	Input/Output	The size in byte of pHardwareVersion buffer

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000130	The device is not in a power state which allows this operation.
0X30000013	The hardware version buffer is not large enough.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.8.5 CMAPI_DevSrv_GetProductType ()

The **CMAPI_DevSrv_GetProductType ()** function retrieves the product type of the device.

Prototype
dword **CMAPI_DevSrv_GetProductType** (dword deviceID, dword* pProductType)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pProductType	Output	Pointer to get the product type in bitmap. In the case of a device with multiple radios, there MAY be multiple settings returned. The bitmap definition follows the definition of RadioType: • 0x00000001: GSM • 0x00000002: WCDMA/UMTS • 0x00000004: CDMA • 0x00000008: EVDO • 0x00000010: TD_SCDMA • 0x00000020: LTE • 0x00000040: WLAN

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.8.6 CMAPI_DevSrv_GetIMSI ()

The **CMAPI_DevSrv_GetIMSI ()** function retrieves the active IMSI(s) from SIM/R-UIM/NAA on UICC.

Prototype
dword **CMAPI_DevSrv_GetIMSI** (dword deviceID, dword systemID, UTF8* pIMSI, dword* pIMSISize, UTF8* pNAAname, dword* pNAAnameSize)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
systemID	Input	The radio system either 3GPP or 3GPP2 to which the function apply when the device is a multi-mode device. • 0x00000000: 3GPP • 0x00000001: 3GPP2
pIMSI	Output	The IMSI.
pIMSISize	Input/Output	The size in byte of pIMSI buffer
pNAAname	Output	NAAname (see CMAPI_DevSrv_GetNAAavailable())
pNAAnameSize	Input/Output	The size in byte of NAAname buffer

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000107	System not supported by the device
0X00000130	The device is not in a power state which allows this operation.
0X30000013	The IMSI buffer is not large enough
0X30000014	The NAA name buffer is not large enough
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.8.7 CMAPI_DevSrv_GetMDN ()

The **CMAPI_DevSrv_GetMDN ()** function retrieves the MDN (only applicable to 3GPP2 systems).

Prototype
dword **CMAPI_DevSrv_GetMDN** (dword deviceID, UTF8* pMDN, dword* pMDNSize)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pMDN	Output	The MDN.
pMDNSize	Input/Output	The size in byte of pMDN buffer

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000108	The requested data is not meaningful for a 3GPP device.
0X00000130	The device is not in a power state which allows this operation.
0X30000015	The MDN buffer is not large enough
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.8.8 CMAPI_DevSrv_GetIMEI ()

The **CMAPI_DevSrv_GetIMEI ()** function retrieves the IMEI (only applicable to 3GPP systems).

Prototype
dword **CMAPI_DevSrv_GetIMEI** (dword deviceID, UTF8* pIMEI, dword* pIMEISize)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pIMEI	Output	The IMEI.
pIMEISize	Input/Output	The size in byte of pIMEI buffer

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000109	The requested data is not meaningful for a 3GPP2 device.
0X00000130	The device is not in a power state which allows this operation.
0X30000016	The IMEI buffer is not large enough
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.8.9 CMAPI_DevSrv_GetESN ()

The **CMAPI_DevSrv_GetESN ()** function retrieves the ESN (only applicable to 3GPP2 systems).

Prototype
dword **CMAPI_DevSrv_GetESN** (dword deviceID, UTF8* pESN, dword* pESNSize)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pESN	Output	The ESN.
pESNSize	Input/Output	The size in byte of pESN buffer

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000108	The requested data is not meaningful for a 3GPP device.
0X00000130	The device is not in a power state which allows this operation.
0X30000017	The ESN buffer is not large enough.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.8.10 CMAPI_DevSrv_GetMEID ()

The **CMAPI_DevSrv_GetMEID ()** function retrieves the MEID (only applicable to 3GPP2 systems).

Prototype
dword **CMAPI_DevSrv_GetMEID** (dword deviceID, UTF8* pMEID, dword* pMEIDSize)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pMEID	Output	The MEID.
pMEIDSize	Input/Output	The size in byte of pMEIDInfo buffer

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000108	The requested data is not meaningful for a 3GPP device.
0X00000130	The device is not in a power state which allows this operation.
0X30000018	The MEID buffer is not large enough
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.8.11 CMAPI_DevSrv_GetMSISDN ()

The **CMAPI_DevSrv_GetMSISDN ()** function retrieves the MSISDN from the active NAA in the SIM/UICC (only applicable to 3GPP systems).

Prototype
dword **CMAPI_DevSrv_GetMSISDN** (dword deviceID, UTF8* pMSISDN, dword* pMSISDNSize)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pMSISDN	Output	The list of MSISDN numbers (see [3GPP TS 51.011] or [3GPP TS 31.102] for details). Each MSISDN number will be separated by character ",". Each field of an MSISDN number (Alpha Identifier, TON value in decimal format, NPI value in decimal format, Dialling Number/SSC value in decimal format...) will be separated by a space. "Length of BCD number/SSC contents", "Capability/Configuration1 Record Identifier" and "Extension1 Record Identifier" are not transmitted.
pMSISDNSize	Input/Output	The size in byte of pMSISDN buffer

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000109	The requested data is not meaningful for a 3GPP2 device.
0X00000130	The device is not in a power state which allows this operation.
0X30000019	The MSISDN buffer is not large enough
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.8.12 CMAPI_DevSrv_GetDeviceStatus ()

The **CMAPI_DevSrv_GetDeviceStatus ()** function retrieves the device status.

Prototype
dword **CMAPI_DevSrv_GetDeviceStatus** (dword deviceID, dword* pDeviceStatus)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pDeviceStatus	Output	Indicates the device status in a bitmap. • 0x00000000: Device unplugged - this value can used for unplugged and for unknown status • 0x00000001: Device plugged but unavailable • 0x00000002: Device plugged and available

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.8.13 CMAPI_DevSrv_GetFirmwareVersion ()

The **CMAPI_DevSrv_GetFirmwareVersion ()** function retrieves the firmware version of the device.

Prototype
dword **CMAPI_DevSrv_GetFirmwareVersion** (dword deviceID, UTF8* pFwVersion, dword* pFwVersionSize)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pFwVersion	Output	The firmware version of the device.
pFwVersionSize	Input/Output	The size in byte of pFwVersion buffer

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000130	The device is not in a power state which allows this operation.
0X3000001A	The FWVersion buffer is not large enough
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.8.14 CMAPI_DevSrv_GetRFSwitch ()

The **CMAPI_DevSrv_GetRFSwitch ()** function retrieves the radio switch status (Radio On / Off).

Prototype
dword **CMAPI_DevSrv_GetRFSwitch** (dword deviceID, dword* pRFStatus)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pRFStatus	Output	Pointer to get the radio switch status in bitmap. In the case of a device with multiple radios, there MAY be multiple settings returned. The bitmap definition follows the definition of RadioType: • 0x00000000: All Radio OFF • 0x00000001: GSM • 0x00000002: WCDMA/UMTS • 0x00000004: CDMA

		• 0x00000008: EVDO • 0x00000010: TD_SCDMA • 0x00000020: LTE • 0x00000040: WLAN

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.8.15 CMAPI_DevSrv_SetRadioState ()

The **CMAPI_DevSrv_SetRadioState ()** function is used to set the radio power state of the device.

Prototype
dword **CMAPI_DevSrv_SetRadioState** (dword deviceID, RadioType Radio, RadioState State)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
Radio	Input	Please see the definition of RadioType
State	Input	Please see the definition of RadioState

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0x00000131	Requested power state is not supported by the device (ex power saving)
0X00000133	The power state is invalid.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated

	credentials to perform this operation.

7.8.16 CMAPI_DevSrv_SetRadioState_Async ()

The **CMAPI_DevSrv_SetRadioState_Async ()** function is used to set the power state of a radio within a device. **CMAPI_DevSrv_SetRadioState_Async** is asynchronous; it initiates a change of the power state and then returns immediately. When the change of the radio power state has finished, the callback **CMAPI_Callback_SetRadioState_Async_Complete** is invoked.

NOTE: Shutting the power of the device completely off may result in an additional callback which indicates a device removal.

Prototype
dword **CMAPI_DevSrv_SetRadioState_Async** (dword deviceID, RadioType Radio, RadioState State)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
Radio	Input	Please see the definition of RadioType
State	Input	Please see the definition of RadioState

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open.
0X00000104	The radio references a radio which the device does not support.
0X00000130	The device is not in a power state which allows this operation.
0X00000131	The device does not support the indicated power state. (ex power saving)
0X00000133	The power state is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.8.17 CMAPI_DevSrv_GetControlKeyStatus ()

The **CMAPI_DevSrv_GetControlKeyStatus ()** function is used to get the specified Mobile Equipment (device) de-personalization control key status.

Prototype
dword **CMAPI_DevSrv_GetControlKeyStatus** (dword deviceID, dword systemID, dword controlKeyID, dword* pcontrolKeyStatus, dword* pverifyRetriesLeft, dword* punblockRetriesLeft)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
systemID	Input	The radio system, either GPP or 3GPP2, to which the function apply when the device is a multi-mode device. • 0x00000000: 3GPP • 0x00000001: 3GPP2
controlKeyID	Input	The ID of the specified Control Key: • 0x00000000: Network Control Key (NCK, only applicable for 3GPP systems) • 0x00000001: Network Subset Control Key (NSCK, only applicable for 3GPP systems) • 0x00000002: Service Provider Control Key (SPCK) • 0x00000003: Corporate Control Key (CCK) • 0x00000004: Personalization Control Key (PCK) • 0x00000005: Network Type 1 Control Key (NCK1, only applicable for 3GPP2 systems) • 0x00000006: Network Type 2 Control Key (NCK2, only applicable for 3GPP2 system) • 0x00000007: HRPD Network Control Key (HNCK, only applicable for 3GPP2 systems) See [3GPP TS 22.022] and [3GPP2 C.S0068] for Control Keys definition and procedures.
pcontrolKeyStatus	Output	Control Key Status: • 0x00000000: Deactivated • 0x00000001: Activated • 0x00000002: Blocked
pverifyRetriesLeft	Output	The number of retries left after which the control key will be blocked
punblockRetriesLeft	Output	The number of unblock retries left after which the control key will be permanently blocked

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open.
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00000134	The system is invalid
0X00000210	Control Key not supported by this system (when an ID of a 3GPP2 only Control Key is sent to a 3GPP system device or when an ID of a 3GPP only Control Key is sent to a 3GPP2 system device).
0X00000211	The control key value is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.8.18 CMAPI_DevSrv_DeactivateControlKey ()

The **CMAPI_DevSrv_DeactivateControlKey ()** function is used to deactivate the specified Mobile Equipment (device) de-personalization control key. Activation of the control key is performed outside the control of the OpenCMAPI.

Prototype
dword **CMAPI_DevSrv_DeactivateControlKey** (dword deviceID, dword systemID, dword controlKeyID, UTF8* controlKeyValue, dword* pVerifyRetriesLeft)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
systemID	Input	The radio system, either 3GPP or 3GPP2, to which the function apply when the device is a multi-mode device. • 0x00000000: 3GPP • 0x00000001: 3GPP2
controlKeyID	Input	The ID of the specified Control Key: • 0x00000000: Network Control Key (NCK, only applicable for 3GPP systems) • 0x00000001: Network Subset Control Key (NSCK, only applicable for 3GPP systems) • 0x00000002: Service Provider Control Key (SPCK) • 0x00000003: Corporate Control Key (CCK) • 0x00000004: Personalization Control Key (PCK) • 0x00000005: Network Type 1 Control Key (NCK1, only applicable for 3GPP2 systems)

		• 0x00000006: Network Type 2 Control Key (NCK2, only applicable for 3GPP2 system) • 0x00000007: HRPD Network Control Key (HNCK, only applicable for 3GPP2 systems) See [3GPP TS 22.022] and [3GPP2 C.S0068] for Control Keys definition and procedures.
controlKeyValue	Input	Control Key de-personalization decimal (Maximum 16 digits)
pVerifyRetriesLeft	Output	The number of retries left after which the control key will be blocked

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open.
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00000134	The system id is invalid
0X00000210	Control Key not supported by this system (when an ID of a 3GPP2 only Control Key is sent to a 3GPP system device or when an ID of a 3GPP only Control Key is sent to a 3GPP2 system device).
0X00000211	The control key value is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.8.19 CMAPI_DevSrv_UnblockControlKey () (Optional)

The **CMAPI_DevSrv_UnblockControlKey ()** function is used to unblock the specified Mobile Equipment (device) de-personalization control key.

Prototype
dword **CMAPI_DevSrv_UnblockControlKey** (dword deviceID, dword systemID, dword controlKeyID, UTF8* controlKeyUnblockValue, dword* punblockRetriesLeft)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
systemID	Input	The radio system either 3GPP or 3GPP2 to which the function apply when the device is a multi-mode device. • 0x00000000: 3GPP • 0x00000001: 3GPP2
controlKeyID	Input	The ID of the specified Control Key:

		• 0x00000000: Network Control Key (NCK, only applicable for 3GPP systems) • 0x00000001: Network Subset Control Key (NSCK, only applicable for 3GPP systems) • 0x00000002: Service Provider Control Key (SPCK) • 0x00000003: Corporate Control Key (CCK) • 0x00000004: Personalization Control Key (PCK) • 0x00000005: Network Type 1 Control Key (NCK1, only applicable for 3GPP2 systems) • 0x00000006: Network Type 2 Control Key (NCK2, only applicable for 3GPP2 system) • 0x00000007: HRPD Network Control Key (HNCK, only applicable for 3GPP2 systems) See [3GPP TS 22.022] and [3GPP2 C.S0068] for Control Keys definition and procedures.
controlKeyUnblockValue	Input	Control Key unblock decimal (Maximum 16 digits)
punblockRetriesLeft	Output	The number of unblock retries left after which the control key will be permanently blocked

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred. Consult the logger for details.
0X00000007	This optional function is not supported by this implementation
0X00000101	The deviceID references a non-existing device or a device which is not open.
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00000134	The system ID is invalid
0X00000210	Control Key not supported by this system (when an ID of a 3GPP2 only Control Key is sent to a 3GPP system device or when an ID of a 3GPP only Control Key is sent to a 3GPP2 system device).
0X00000211	The control key unblock value is not valid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.9 PINs/PUKs Management APIs

7.9.1 Access Control

The control mechanism described in the "Access Control" chapter of the "UICC Management APIs" chapter also applies to this section.

For Mobile Broadband devices, the implementation of the Access Control function is optional in accordance with "UICC Management APIs" chapter.

For all other devices, all the functions described in the current chapter entitled "PINs/PUKs Management APIs" SHALL be implemented.

7.9.2 CMAPI_DevSrv_GetNAAavailable ()

The **CMAPI_DevSrv_GetNAAavailable ()** function is used to get all the available NAAs and the corresponding Application labels.

Prototype
dword **CMAPI_DevSrv_GetNAAavailable** (dword deviceID, NAANametype* pNAAList, dword* pNAAListSize, dword* pNAAListCount)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pNAAList	Output	See NAANameType. The NAAList structures will be laid out at the front of the buffer.
pNAAListSize	Input/Output	The number of bytes in the NAA List buffer or if insufficient contains the necessary size.
pNAAListCount	Output	The number of elements in the array pointed by the pNAAname

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000130	The device is not in a power state which allows this operation.
0X30001000	The size for the pNAAlist buffer is not sufficient; the NAAListsize will contain the number of bytes required.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.9.3 CMAPI_DevSrv_EnablePIN ()

The **CMAPI_DevSrv_EnablePIN ()** function is used to enable PIN protection.

Prototype
dword **CMAPI_DevSrv_EnablePIN** (dword deviceID, byte PINType, UTF8* PINCode, UTF8* NAAname, byte* pRetry)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
PINType	Input	The type of the PIN: 0—PIN, 1—PIN2
PINCode	Input	PIN code, value '0' ~ '9', 4-8 digit length.
NAAname	Input	NAA name to indicate which PIN will be operated NAA name can be: SIM, R-UIM, USIM_1, USIM_2, ..., USIM_N, CSIM_1, CSIM_2, ..., CSIM_N, ISIM_1, ISIM_2, ..., ISIM_N. If there is no NAA name from the previous list to be associated to one or several AID values available into the UICC (see [ETSI TS 102 221]), then the AID value shall be put in this field.
pRetry	Output	Number of attempts left

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X1001SW1SW2	Wrong PIN.
0X1002SW1SW2	PIN is blocked. PUK (UNBLOCK PIN) needed.
0X1007SW1SW2	Invalid parameter(s)
0X11000001	The NAA Name is invalid
0X11000002	The PIN Type is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.9.4 CMAPI_DevSrv_DisablePIN ()

The **CMAPI_DevSrv_DisablePIN ()** function is used to disable PIN protection.

Prototype
dword **CMAPI_DevSrv_DisablePIN** (dword deviceID, byte PINType, UTF8* PINCode, UTF8* NAAname, byte* pRetry)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
PINType	Input	The type of PIN: 0—PIN, 1—PIN2
PINCode	Input	PIN code, value '0' ~ '9', 4-8 digit length.
NAAname	Input	NAA name to indicate which PIN will be operated NAA name can be: SIM, R-UIM, USIM_1, USIM_2, ..., USIM_N, CSIM_1, CSIM_2, ..., CSIM_N, ISIM_1, ISIM_2, ..., ISIM_N. If there is no NAA name from the previous list to be associated to one or several AID values available into the UICC (see [ETSI TS 102 221]), then the AID value shall be put in this field.
pRetry	Output	Number of attempts left

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000104	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X1001SW1SW2	Wrong PIN.
0X1002SW1SW2	PIN is blocked. PUK (UNBLOCK PIN) needed.
0X1007SW1SW2	Invalid parameter(s)
0X11000001	The NAA Name is invalid
0X11000002	The PIN Type is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.9.5 CMAPI_DevSrv_VerifyPIN ()

The **CMAPI_DevSrv_VerifyPIN ()** function is used to verify a PIN.

Prototype
dword **CMAPI_DevSrv_VerifyPIN** (dword deviceID, byte PINType, UTF8* PINCode, UTF8* NAAname, byte* pRetry)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
PINType	Input	The type of PIN: 0—PIN, 1—PIN2
PINCode	Input	PIN code, value '0' ~ '9', 4-8 digit length.
NAAname	Input	NAA name to indicate which PIN will be operated NAA name can be: SIM, R-UIM, USIM_1, USIM_2, ..., USIM_N, CSIM_1, CSIM_2, ..., CSIM_N, ISIM_1, ISIM_2, ..., ISIM_N. If there is no NAA name from the previous list to be associated to one or several AID values available into the UICC (see [ETSI TS 102 221]), then the AID value shall be put in this field.
pRetry	Output	Number of attempts left

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X1001SW1SW2	Wrong PIN.
0X1002SW1SW2	PIN is blocked. PUK (UNBLOCK PIN) needed.
0X1007SW1SW2	Invalid parameter(s)
0X11000001	The NAA Name is invalid
0X11000002	The PIN Type is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.9.6 CMAPI_DevSrv_UnblockPIN ()

The **CMAPI_DevSrv_UnblockPIN ()** function is used to unblock a PIN.

Prototype
dword **CMAPI_DevSrv_UnblockPIN** (dword deviceID, byte PUKType, UTF8* PUK, UTF8* NewPINCode, UTF8* NAAname, byte* pRetry)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
PUKType	Input	The type of PUK: 0—PUK, 1—PUK2
PUK	Input	PUK code, value '0' ~ '9', 8 digit length.
PINCode	Input	New PIN code, value '0' ~ '9', 4-8 digit length.
NAAname	Input	NAA name to indicate which PIN will be operated NAA name can be: SIM, R-UIM, USIM_1, USIM_2, ..., USIM_N, CSIM_1, CSIM_2, ..., CSIM_N, ISIM_1, ISIM_2, ..., ISIM_N. If there is no NAA name from the previous list to be associated to one or several AID values available into the UICC (see [ETSI TS 102 221]), then the AID value shall be put in this field.
pRetry	Output	Number of attempts left

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X1005SW1SW2	Wrong PUK.
0X1006SW1SW2	PUK (UNBLOCK PIN) blocked.
0X1007SW1SW2	Invalid parameter(s)
0X11000001	The NAA Name is invalid
0X11000003	The PUK Type is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.9.7 CMAPI_DevSrv_ChangePIN ()

The **CMAPI_DevSrv_ChangePIN ()** function is used to change a PIN.

Prototype
dword **CMAPI_DevSrv_ChangePIN** (dword deviceID, byte PINType, UTF8* OldPINCode, UTF8* NewPINCode, UTF8* NAAname, byte* pRetry)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
PINType	Input	The type of PIN: 0—PIN, 1—PIN2
OldPINCode	Input	Old PIN code, value '0' ~ '9', 4-8 digit length.
NewPINCode	Input	New PIN code, value '0' ~ '9', 4-8 digit length.
NAAname	Input	NAA name to indicate which PIN will be operated NAA name can be: SIM, R-UIM, USIM_1, USIM_2, ..., USIM_N, CSIM_1, CSIM_2, ..., CSIM_N, ISIM_1, ISIM_2, ..., ISIM_N. If there is no NAA name from the previous list to be associated to one or several AID values available into the UICC (see [ETSI TS 102 221]), then the AID value shall be put in this field.
pRetry	Output	Number of attempts left

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X1003SW1SW2	Wrong Old PIN.
0X1004SW1SW2	Old PIN is blocked. PUK (UNBLOCK PIN) needed.
0X1007SW1SW2	Invalid parameter(s)
0X11000001	The NAA Name is invalid
0X11000002	The PIN Type is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.10 UICC Management APIs

7.10.1 Access Control

The OpenCMAPI SHALL control the access of Connection Manager applications sending APDUs to the Smart Card (SIM/R-UIM/NAA on UICC) through the Access Control mechanism defined in [GP, SE Access Control] except that the access to UICC is granted for Connection Manager applications if neither the ARA-M nor the ARF is present on the Smart Card. The control SHALL apply for a given Connection Manager application as soon as **CMAPI_API_Open ()** has been called and until the call of **CMAPI_API_Close ()**.

The Smart Card (SIM/R-UIM/NAA on UICC) SHALL be compliant with [GP, SE Access Control] in order to provide the interface to the Access Control mechanism in the device to retrieve the Access Rules.

The Smart Card (SIM/R-UIM/NAA on UICC) issuer SHALL provision Access Rules into the Smart Card according to its security policy as defined in [GP, SE Access Control].

For Mobile Broadband devices, the implementation of the Access Control function and the functions described in the following sub-sections of the current chapter entitled “UICC Management APIs” are optional. However, if these latter functions are implemented the Access Control function SHALL also be implemented.

For all other devices, all the functions described in the current chapter entitled “UICC Management APIs” SHALL be implemented.

7.10.2 CMAPI_UICC_GetTerminalProfile ()

The device SHALL support the class “s”, “Support of CAT over the modem interface”, as specified in [ETSI TS 102 223].

The **CMAPI_UICC_GetTerminalProfile ()** function is used for the Connection Manager Application to get the last TERMINAL PROFILE sent by the device to the SIM/R-UIM/UICC.

Prototype
dword **CMAPI _UICC_GetTerminalProfile** (dword deviceID, byte pTerminalProfile[256])

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pTerminalProfile	Output	The hexadecimal value of the TERMINAL PROFILE as specified in the chapter “Structure and coding of the TERMINAL PROFILE” of [ETSI TS 102 223] for the core part, in the chapter “Structure and coding of the TERMINAL PROFILE” of [3GPP TS 31.111] for the 3GPP specific part, in the chapter “Structure and coding of the TERMINAL PROFILE” of [3GPP2 C.S0035] for the 3GPP2 specific part.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred. Consult the logger for details.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.

0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.10.3 CMAPI_UICC_SetTerminalProfile ()

The device SHALL support the class “s”, “Support of CAT over the modem interface”, as specified in [ETSI TS 102 223].

The **CMAPI_UICC_SetTerminalProfile ()** function is used to transmit to the SIM/R-UIM/UICC via the device the ToolKit functions (i.e.: the TERMINAL PROFILE) that are supported by the Connection Manager Applications.

If several Connection Manager Applications are running in parallel, the Connection Manager API shall verify that there is no overlap between the TERMINAL PROFILE sent by the device and by each of the Connection Manager Applications as specified in [ETSI TS 102 223] (see normative annex). If an overlap exists the Connection Manager API shall send a return value identifying the overlapping ToolKit functions. If an overlap exists between several Connection Manager Applications, the ToolKit functions of the first Connection Manager Application having sent a **CMAPI_UICC_SetTerminalProfile ()** will take precedence over the overlapping ToolKit functions of the other Connection Manager Applications.

The device SHALL combine the facilities provided by the device and the facilities provided by the Connection Manager Applications (also called CAT clients within the Connected Entity in [ETSI TS 102 223]) as specified in [ETSI TS 102 223] before sending the combined TERMINAL PROFILE to the SIM/R-UIM/UICC.

Prototype
dword **CMAPI_UICC_SetTerminalProfile** (dword deviceID, byte terminalProfile[256], byte pOverlappingToolkit[256])

Parameters

Field Name	Mode	Description
deviceID	Input	The ID of the device concerned
terminalProfile	Input	The hexadecimal value of the TERMINAL PROFILE as specified in the chapter “Structure and coding of the TERMINAL PROFILE” of [ETSI TS 102 223] for the core part, in the chapter “Structure and coding of the TERMINAL PROFILE” of [3GPP TS 31.111] for the 3GPP specific part, in the chapter “Structure and coding of the TERMINAL PROFILE” of [3GPP2 C.S0035] for the 3GPP2 specific part.
pOverlappingToolKit	Output	(optional) Overlapping ToolKit function - The hexadecimal value of the TERMINAL PROFILE corresponding only to the overlapping Toolkit functions. This field is only present with Return value 0X00000554”.

Return Values

Value	Description
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00000553	The terminal profile is invalid
0X00000554	The function succeeded except for the overlapping ToolKit functions with the

	device or another or other Connection Manager Application(s)
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.10.4 CMAPI_UICC_SendToolKitEnvelopeCommand ()

The device SHALL support the class “s”, “Support of CAT over the modem interface”, as specified in [ETSI TS 102 223].

The **CMAPI_UICC_SendToolKitEnvelopeCommand ()** function is used for the Connection Manager Application to transmit to the SIM/R-UIM/UICC via the device any ToolKit ENVELOPE command that is supported by the Connection Manager Application and for which no overlapping was identified (see **CMAPI_UICC_SetTerminalProfile ()** and [ETSI TS 102 223]).

Prototype
dword **CMAPI_UICC_SendToolKitEnvelopeCommand** (dword deviceID, byte envelopeCommand[256])

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
envelopeCommand	Input	The hexadecimal value of the ENVELOPE Command as specified in the chapter “ENVELOPE Commands” of [ETSI TS 102 223] for the core part, in the chapter “ENVELOPE Commands” of [3GPP TS 31.111] for the 3GPP specific part, in the chapter “ENVELOPE Commands” of [3GPP2 C.S0035] for the 3GPP2 specific part.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00000551	ENVELOPE command was not sent to SIM/R-UIM/UICC as overlapping was detected.
0X00000552	The envelope command is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.10.5 CMAPI_UICC_SendTerminalResponse ()

The device SHALL support the class “s”, “Support of CAT over the modem interface”, as specified in [ETSI TS 102 223].

The **CMAPI_UICC_SendTerminalResponse ()** function is used for the Connection Manager Application to send a TERMINAL RESPONSE to the SIM/R-UIM/UICC via the device answering to any ToolKit Proactive Command received via the Callback **CMAPI_UICC_ToolKitProactiveCommand** (see callback chapter).

Prototype
dword **CMAPI_UICC_SendTerminalResponse** (dword deviceID, byte terminalResponse[256])

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
terminalResponse	Input	The hexadecimal value of the TERMINAL RESPONSE as specified in the chapter “Structure and coding of the TERMINAL RESPONSE” of [ETSI TS 102 223] for the core part, in the chapter “Structure and coding of the TERMINAL RESPONSE” of [3GPP TS 31.111] for the 3GPP specific part, in the chapter “Structure and coding of the TERMINAL RESPONSE” of [3GPP2 C.S0035] for the 3GPP2 specific part.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00000555	The terminal response is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.11 WLAN APIs

7.11.1 CMAPI_WLAN_IsSupported ()

The **CMAPI_WLAN_IsSupported ()** function is used to determine if WLAN functionality is supported

Prototype
dword **CMAPI_WLAN_IsSupported** (dword deviceID, dword* pWlanSupport)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pWlanSupport	Output	Indicates WLAN support: • 0x00000001: WLAN Supported • 0x00000002: WLAN NOT supported (Device do not support WLAN capability) • 0x00000003: WLAN NOT supported (other reason)

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.11.2 CMAPI_WLAN_AddKnownNetwork ()

The **CMAPI_WLAN_AddKnownNetwork ()** function is used to add a network to the known network list.

Prototype
dword **CMAPI_WLAN_AddKnownNetwork** (dword index, WLANNetwork* pNetwork)

Parameters		
Field Name	**Mode**	**Description**
index	Input	The zero based index which describes the position of the network in the known networks list. Any existing subsequent entry will have their previous index adjusted to be one larger.
pNetwork	Input	The network to add to the known networks list.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00010004	The SSID is invalid
0X00010005	The BSSID is invalid
0X00010006	The Friendly Name is invalid
0X00010007	The security parameter is invalid
0X00010008	The mode parameter is invalid
0X00010009	The hidden parameter is invalid
0X0001000A	The key is invalid
0X0001000B	The EAP authentication method is invalid
0X0001000C	The EAP configuration is invalid
0X00013007	The specified index is to large and would leave a gap in the known networks list
0X00013008	Index is not valid for user defined networks. Please try a higher index.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.11.3 CMAPI_WLAN_UpdateKnownNetwork ()

The **CMAPI_WLAN_UpdateKnownNetwork ()** function is used to update an existing known network record.

Prototype
dword **CMAPI_WLAN_UpdateKnownNetwork** (dword index, WLANNetwork* pNetwork)

Parameters		
Field Name	**Mode**	**Description**
index	Input	The zero based index which describes the position of the network in the known networks list.
pNetwork	Input	The updated network info to reside at the index in the known networks list.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00010001	No network exists at the specified index.
0X00010002	Predefined networks are not able to be modified.
0X00010004	The SSID is invalid

0X00010005	The BSSID is invalid
0X00010006	The Friendly Name is invalid
0X00010007	The security parameter is invalid
0X00010008	The mode parameter is invalid
0X00010009	The hidden parameter is invalid
0X0001000A	The key is invalid
0X0001000B	The EAP authentication method is invalid
0X0001000C	The EAP configuration is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.11.4 CMAPI_WLAN_DeleteKnownNetwork ()

The **CMAPI_WLAN_DeleteKnownNetwork ()** function is used to remove the entry from the known networks list at the specified index.

Prototype
dword **CMAPI_WLAN_DeleteKnownNetwork** (dword index)

Parameters		
Field Name	**Mode**	**Description**
index	Input	The index of the record to remove from the known networks list. Any subsequent records will have their index decremented.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00010001	No network exists at the specified index.
0X00010002	Predefined networks are not able to be modified
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.11.5 CMAPI_WLAN_GetKnownNetwork ()

The **CMAPI_WLAN_GetKnownNetwork ()** function is used to retrieve the known network record information

Prototype
dword **CMAPI_WLAN_GetKnownNetwork** (dword index, WLANNetwork* pNetwork, dword* pNetworkSize)

Parameters		
Field Name	**Mode**	**Description**
index	Input	The index of the known network to retrieve.
pNetwork	Output	The known network record.
pNetworkSize	Input/Output	The size of the structure WLAN network structure

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00010001	No network exists at the specified index.
0X00010003	The size of the network structure is not large enough pNetworkSize contains the minimum size required.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.11.6 CMAPI_WLAN_GetScanResults ()

The **CMAPI_WLAN_GetScanResults ()** function is used to retrieve the list of available WLAN networks. Invoking this call does not force an operation on the device like scanning; it simply retrieves the most recent scan list.

Prototype
dword **CMAPI_WLAN_GetScanResults** (dword deviceID, WLANNetwork* pScanList, dword* pScanListSize, dword* pScanListCount)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pScanList	Output	The buffer to hold the scan list entry
pScanListSize	Input/Output	Contains the number of bytes of the ScanList buffer on input. If buffer size is not sufficient, this will contain the number of bytes needed in the structure on return.
pScanListCount	Output	The number of entries in the scan list

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X30000009	The buffer is insufficient. pScanListSize contains the minimum number of bytes necessary to hold the scan list.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.11.7 CMAPI_WLAN_Scan_Async ()

The **CMAPI_WLAN_Scan_Async ()** function is used to initiate a scan for WLAN networks. This initiates an asynchronous process to discover WLAN networks available. The calling thread returns immediately. The result is reported in callback **CMAPI_Callback_ScanWLANComplete ()**.

Prototype
dword **CMAPI_WLAN_Scan_Async** (dword deviceID, dword Timeout)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
Timeout	Input	The maximum time for the WLAN network scan (in seconds).

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.11.8 CMAPI_WLAN_Connect ()

The **CMAPI_WLAN_Connect ()** function is used to connect to a WLAN network. This operation occurs asynchronously.

Prototype
dword **CMAPI_WLAN_Connect** (dword deviceID, WLANNetwork* pNetwork, dword associationTimeout, dword grantTimeout)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pNetwork	Input	Specifies the network to connect.
associationTimeout	Input	Specifies the number of milliseconds to allow an association to the network to be setup before reporting failure.
grantTimeout	Input	Specifies the number of milliseconds to allow a DHCP operation to proceed before reporting failure.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00010004	The SSID is invalid
0X00010005	The BSSID is invalid
0X00010006	The Friendly Name is invalid
0X00010007	The security parameter is invalid
0X00010008	The mode parameter is invalid
0X00010009	The hidden parameter is invalid
0X0001000A	The key is invalid
0X0001000B	The EAP authentication method is invalid
0X0001000C	The EAP configuration is invalid
0X0001000D	The WLAN Encryption Type is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.11.9 CMAPI_WLAN_ConnectKnownNetwork ()

The **CMAPI_WLAN_ConnectKnownNetwork ()** function is used to connect to a WLAN network in the known networks list. This operation occurs asynchronously.

Prototype
dword **CMAPI_WLAN_ConnectKnownNetwork** (dword deviceID, UTF8* SSID, UTF8* BSSID, dword associationTimeout, dword grantTimeout)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
SSID	Input	The SSID of the known network
BSSID	Input	(Optional) The BSSID of the known network
associationTimeout	Input	Specifies the number of milliseconds to allow an association to the network to be setup before reporting failure.
grantTimeout	Input	Specifies the number of milliseconds to allow a DHCP operation to proceed before reporting failure.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X0001000D	The WLAN Encryption Type is invalid
0X00012001	The SSID does not reference a valid known network.
0X00012002	The BSSID does not reference a valid known network
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.11.10 CMAPI_WLAN_Disconnect ()

The **CMAPI_WLAN_Disconnect ()** function is used to disconnect any connected WLAN network.

Prototype
dword **CMAPI_WLAN_Disconnect** (dword deviceID)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00011001	There is no existing WLAN connection
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.11.11 CMAPI_WLAN_GetConnectionMode ()

The **CMAPI_WLAN_GetConnectionMode ()** function is used to determine if connectivity is being actively sought by the enabler or if manual connection requests are required.

Prototype
dword **CMAPI_WLAN_GetConnectionMode** (dword deviceID, dword* pMode)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pMode	Output	Indicates connectivity mode. • 0x00000001: Auto connect to known networks • 0x00000002: Manual connect (known and unknown networks) • 0x00000003: Manual connect (only to known networks – subject to some policies)

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.11.12 CMAPI_WLAN_SetConnectionMode ()

The **CMAPI_WLAN_SetConnectionMode ()** function is used to change the connectivity mode. Changing connectivity mode will not affect any established connection.

Prototype
dword **CMAPI_WLAN_SetConnectionMode** (dword deviceID, dword mode)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
mode	Input	Indicates connectivity mode. • 0x00000001: Auto connect to known networks • 0x00000002: Manual connect • 0x00000003: Manual connect (only to known networks)

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00011002	Security mode does not allow connectivity to unknown networks.
0X00013009	The mode is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.11.13 CMAPI_WLAN_ResetDevice ()

The **CMAPI_WLAN_ResetDevice ()** function is used to reset the device. This causes the device to be power cycled.

Prototype
dword **CMAPI_WLAN_ResetDevice** (dword deviceID)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned

Return Values	
Value	**Description**

0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.11.14 CMAPI_WLAN_GetConnectedParameters ()

The **CMAPI_WLAN_GetConnectedParameters ()** function is used to retrieve values related to the associated network.

Prototype
dword **CMAPI_WLAN_GetConnectedParameters** (dword deviceID, ConnectedParameters* pParameters, dword* pParametersSize, UTF8* pMacAddress, dword* pMacAddressSize)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pParameters	Output	The ip address, mask, proxy information
pParametersSize	Input/Output	The size of the pParameters buffer in bytes
pMacAddress	Output	The physical address of the access point
pMacAddressSize	Input/Output	The size of the pMacAddress buffer in bytes

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X30000020	The pParameters buffer is not large enough. pParametersSize contains the minimum buffer length required.
0X30000021	The pMacAddress buffer is not large enough. pMacAddressSize contains the minimum buffer length required.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.11.15 CMAPI_WLAN_SetConnectedParameters ()

The **CMAPI_WLAN_SetConnectedParameters ()** function is used to set various attributes of an existing connection.

Prototype
dword **CMAPI_WLAN_SetConnectedParameters** (dword deviceID, ConnectedParameters* pParameters)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pParameters	Input	The parameters to set.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00011005	Operation is prohibited by security policy.
0X0001300A	The address is invalid
0X0001300B	The subnet mask is invalid
0X0001300C	The http proxy is invalid
0X0001300D	The mac address is invalid
0X0001300E	The default gateway is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.11.16 CMAPI_WLAN_CancelOperation ()

The **CMAPI_WLAN_CancelOperation ()** function is used to cancel any pending operation like connect or scan.

Prototype
dword **CMAPI_WLAN_CancelOperation** (dword deviceID)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00011006	No pending operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.11.17 CMAPI_WLAN_ConnectWPS ()

The **CMAPI_WLAN_ConnectWPS ()** function is used to initiate a connection with the WPS button push method.

Prototype
dword **CMAPI_WLAN_ConnectWPS** (dword deviceID)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.

0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.11.18 CMAPI_WLAN_ConnectPinWPS ()

The **CMAPI_WLAN_ConnectPinWPS ()** function is used to initiate a connection with the WPS pin method.

Prototype
dword **CMAPI_WLAN_ConnectPinWPS** (dword deviceID, byte* Pin, dword Pinlength)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
Pin	Input	The pin entered by the user in hexadecimal.
Pinlength	Input	The length of the pin provided in bytes.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00011007	The pin for WPS was malformed or incorrect size
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.11.19 CMAPI_WLAN_ConnectionState ()

The **CMAPI_WLAN_ConnectionState ()** function is used to determine if WLAN is connected.

Prototype
dword **CMAPI_WLAN_ConnectionState** (dword deviceID, dword* pStatus)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
pStatus	Output	Indicates WLAN connection status. • 0x00000000: Connection attempt starting • 0x00000001: Attempting association • 0x00000002: Association failed

		• 0x00000003: Attempting authentication • 0x00000004: Authentication failed • 0x00000005: Requesting IP address • 0x00000006: IP grant failed • 0x00000010: Connected • 0x00000020: Disconnecting • 0x00000021: Disconnected

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.11.20 CMAPI_WLAN_SearchNetwork_Async ()

The **CMAPI_WLAN_SearchNetwork_Async ()** function is used to check the availability of a specific WLAN network. The calling thread returns immediately. This operation occurs asynchronously. The result is reported in callback **CMAPI_Callback_SearchWLANNetworkComplete ()**.

Prototype
dword **CMAPI_WLAN_SearchNetwork_Async** (dword deviceID, dword Timeout, WLANNetwork* pNetwork)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
Timeout	Input	The maximum time for the search for the WLAN Network (in seconds).
pNetwork	Input	The network to search for

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.

0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.12 Statistics APIs

7.12.1 CMAPI_NetStatistic_GetConnectionStatistics ()

The **CMAPI_NetStatistic_GetConnectionStatistics ()** function is used to obtain network traffic statistics info

Prototype
dword **CMAPI_NetStatistic_GetConnectionStatistics** (dword deviceID, dword CellularProfileID, qword* pTx, qword* pRx, qword* pAverageTx, qword* pAverageRx, qword* pMaxTx, qword* pMaxRx, qword* pDuration, dword* pOverflow)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
CellularProfileID	Input	Optional - The ID of the Cellular Profile to be used for this function. 0xFFFFFFFF is used in the optional condition.
pTx	Output	Bytes sent for a given connection
pRx	Output	Bytes received for a given connection
pAverageTx	Output	Average upload speed in Bit/s for the given connection
pAverageRx	Output	Average download speed in Bit/s for the given connection
pMaxTx	Output	Maximum upload speed in Bit/s for the given connection
pMaxRx	Output	Maximum download speed in Bit/s for the given connection
pDuration	Output	The connection duration in milliseconds
pOverflow	Output	Bitmap parameter to signal overflow argument • 0x00000001: Tx overflow • 0x00000002: Rx overflow • 0x00000004: duration overflow

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000111	The device is not connected
0X00002001	The Cellular profile ID does not exist
0XF0000001	The security request supplied when the API was opened does not grant privilege to

	access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.13 Information Status APIs

7.13.1 CMAPI_Information_GetPINStatus ()

The **CMAPI_Information_GetPINStatus ()** function is used to return the status of the PINs and PUKs of all active SIM/R-UIM/NAA on UICC for a dedicated device.

Prototype
dword **CMAPI_Information_GetPINStatus** (dword deviceID, PINPUKStatustype* pPINPUKStatusList, dword* pPINPUKStatusListSize, dword* pPINPUKStatusListCount)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pPINPUKStatusList	Output	The status of the PINs/PUKs for all active NAAs (see PINPUKStatusType definition). The PINPUKStatus structures will be laid out at the front of the buffer.
pPINPUKStatusListSize	Input/Output	The size of the pPINPUKStatusList buffer or if insufficient contains the necessary size.
pPINPUKStatusListCount	Output	Contains the number of entries in the pPINPUKStatusList

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000121	The device does not offer this capability
0X00000130	The device is not in a power state which allows this operation.
0X30000022	The pPINPUKStatusList is not large enough.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.13.2 CMAPI_Information_GetNetworkSelectionMode ()

The **CMAPI_Information_GetNetworkSelectionMode ()** function is used to determine the network selection mode.

Prototype
dword **CMAPI_Information_GetNetworkSelectionMode** (dword deviceID, RadioType Radio, dword* pState)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
Radio	Input	See RadioType definition
pState	Output	The state of the network selection mode: • 0x00000000: Automatic (Manual operator selection permitted) • 0x00000001: Manual (Manual operator selection active, may return to Automatic)

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000105	The radio references a radio which the device does not support.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.13.3 CMAPI_Information_GetSignalStrength ()

The **CMAPI_Information_GetSignalStrength ()** function is used obtain the current signal strength value, the percentage of signal present and the signal quality.

Prototype
dword **CMAPI_Information_GetSignalStrength** (dword deviceID, RadioType Radio, dword* pSignalStrengthRaw, dword* pSignalStrengthPercent, dword* pSignalQualityPercent)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
Radio	Input	See RadioType definition
pSignalStrengthRaw	Output	The signal strength value in dBm

pSignalStrengthPercent	Output	The signal strength as a percentage - SHOULD be adjusted to device capabilities.
pSignalQualityPercent	Output	The signal quality as a percentage - SHOULD be adjusted to device capabilities.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000105	The radio references a radio which the device does not support.
0X00000130	The device is not in a power state which allows this operation.
0X00006003	Remote system not present
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.13.4 CMAPI_Information_GetCSNetworkRegistration ()

The **CMAPI_Information_GetCSNetworkRegistration ()** function is used to determine if a circuit switched registration is present.

Prototype
dword **CMAPI_Information_GetCSNetworkRegistration** (dword deviceID, RadioType Radio, byte* pState)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
Radio	Input	See RadioType definition
pState	Output	Indicates if a circuit switched registration is present • 0x00: Not Registered • 0x01: Registered

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.

0X00000105	The radio references a radio which the device does not support.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.13.5 CMAPI_Information_GetPSNetworkRegistration ()

The **CMAPI_Information_GetPSNetworkRegistration ()** function is used to determine if a packet switched attachment is present.

Prototype
dword **CMAPI_Information_GetPSNetworkRegistration** (dword deviceID, RadioType Radio, byte* pState)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
Radio	Input	See RadioType definition
pState	Output	Indicates if a packet switched attachment is present • 0x00: Not attached • 0x01: Attached

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000105	The radio references a radio which the device does not support.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.13.6 CMAPI_Information_GetAPN ()

The **CMAPI_Information_GetAPN ()** function is to obtain the APN identifier.

To iterate through the supplied APNs, the caller would start at the 0 index and monotonically increment the index until the error code indicates there are no more records available.

The APN is defined in [3GPP TS 23.003] as of consisting of a mandatory Network Identifier and an optional Operator Identifier.

Prototype
dword **CMAPI_Information_GetAPN** (dword deviceID, RadioType Radio, dword CellularProfileID, dword index, UTF8* pNetworkIdentifier, dword* pNetworkIdentifierSize, UTF8* pOperatorIdentifier, dword* pOperatorIdentifierSize)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
Radio	Input	See RadioType definition
CellularProfileID	Input	Optional - The ID of the Cellular Profile to be used for this function. 0xFFFFFFFF is used in the optional condition.
index	Input	The index of the entry to return (0xFFFFFFFF returns the current APN in use)
pNetworkIdentifier	Output	The network identifier
pNetworkIdentifierSize	Input/Output	The size of the network identifier buffer
pOperatorIdentifier	Output	The operator identifier
pOperatorIdentifierSize	Input/Output	The size of the operator identifier buffer

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation
0X00000105	The radio references a radio which the device does not support.
0X00000130	The device is not in a power state which allows this operation.
0X00002001	The Cellular profile ID does not exist
0X00006004	The supplied index identifies a record which does not exist.
0X00006005	Current APN cannot be retrieved because there is no connection.
0X30000004	The network identifier buffer is not large enough, pNetworkIdentifierSize holds the minimum necessary size in bytes
0X30000005	The operator identifier buffer is not large enough, pOperatorIdentifierSize holds the minimum necessary size in bytes.

0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.13.7 CMAPI_Information_GetIPAddress ()

The **CMAPI_Information_GetIPAddress ()** function is used to retrieve the current IP address assigned to the device and the type of the address assigned.

Prototype
dword **CMAPI_Information_GetIPAddress** (dword deviceID, dword CellularProfileID, dword addressType, IPAddress* pAddress, dword* pAddressSize)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
CellularProfileID	Input	The ID of the Cellular Profile to be used for this function If the ID is set to "0xFFFFFFFF", the IPAddress of the WLAN interface is returned
addressType	Input	The types of IP Address to return: • 0x00000001: IPv4 • 0x00000002: IPv6 (IPv4-compatible IPv6 Address in case of IPv4 address) • 0x00000003: IPv6 (IPv4-mapped IPv6 address in case of IPv4 address)
pAddress	Output	The address for the current connection
pAddressSize	Input/Output	The address size

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000014	Not connected
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000105	The radio references a radio which the device does not support.
0X00000130	The device is not in a power state which allows this operation.
0X00002001	The Cellular profile ID does not exist
0X00006006	The type of IP address is not available.
0X00006007	IP Address is not currently assigned (advisable to retry call)
0X00006008	Authentication failure
0X30000024	The address buffer is not large enough, pAddressSize contains the minimum required size in bytes.

0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.13.8 CMAPI_Information_GetRoamingStatus ()

The **CMAPI_Information_GetRoamingStatus ()** function is used to retrieve the current roaming status.

Prototype
dword **CMAPI_Information_GetRoamingStatus** (dword deviceID, dword systemID, dword* pState)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
systemID	Input	The radio system either 3GPP or 3GPP2 to which the function apply when the device is a multi-mode device. • 0x00000000: 3GPP • 0x00000001: 3GPP2
pState	Output	Indication of the roaming state • 0x000000000: Home • 0x000000001: Roaming

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation
0X00000107	System not supported by the device
0X00000130	The device is not in a power state which allows this operation.
0X00006003	Remote system is not present.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.13.9 CMAPI_Information_GetDriverVersion ()

The **CMAPI_Information_GetDriverVersion ()** function is used to retrieve the driver version.

Prototype
dword **CMAPI_Information_GetDriverVersion** (dword deviceID, UTF8* pDriverVersion, dword* pDriverVersionSize)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pDriverVersion	Output	Indicates the driver version number
pDriverVersionSize	Input/Output	The size of the data

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000130	The device is not in a power state which allows this operation.
0X30000025	Version buffer is not large enough, pDriverVersionSize contains the required size in bytes.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.13.10 CMAPI_Information_GetRATType ()

The **CMAPI_Information_GetRATType ()** function is used to retrieve the radio access technology.

Prototype
dword **CMAPI_Information_GetRATType** (dword deviceID, RadioType Radio, dword* pTypes)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
Radio	Input	See RadioType definition
pTypes	Output	Indication of the radio access technology currently used In the case of a device with multiple radios, there MAY be multiple settings returned. • 0x00000010: GSM service • 0x00000020: GPRS service • 0x00000040: EDGE service • 0x00000080: CDMA service • 0x00000100: QNC service • 0x00000200: 1X-RTT service • 0x00000400: EV-DO service • 0x00000800: EV-DV service

		• 0x00001000: IOTA service • 0x00002000: IOTA REVA service • 0x00004000: UMTS service • 0x00008000: HSDPA service (Included for legacy purpose, not all operators use HSDPA+) • 0x00010000: HSUPA service • 0x00020000: HSPA Plus service • 0x00040000: PHS service • 0x00080000: FOMA service • 0x00100000: LTE service • 0x10000000: WLAN service

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000105	The radio references a radio which the device does not support.
0X00000130	The device is not in a power state which allows this operation.
0X00006003	Remote system is not present
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.13.11 CMAPI_Information_GetQoS ()

The **CMAPI_Information_GetQoS ()** function is used to retrieve the QoS parameters related to the network as defined in [3GPPTS 23.107].

Prototype
dword **CMAPI_Information_GetQoS** (dword deviceID,dword CellularProfileID, QoSStructure* pQoSContextList, dword* pQoSContextListSize, dword* pQoSContextListCount)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
CellularProfileID	Input	Cellular Profile ID, the unique identity for a profile. 0xFFFFFFFF is reserved and cannot be used.
pQoSContextList	Output	The list of the QoS structures per context associated with the Cellular Profile ID
pQoSContextListSize	Input/Out	The size of the buffer in Bytes for the QoScontextlist or if

	put	insufficient, contain the necessary size
pQoSContextListSizeCount	Output	Number of entries in the list.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000013	QoS unsupported
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00002001	The Cellular profile ID does not exist
0X00006003	Remote system not present
0X30000026	The pQoSContextList Buffer is not large enough.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.13.12 CMAPI_Information_GetWLANConnection ()

The **CMAPI_Information_GetWLANConnection ()** function is used to retrieve identifying data of the currently connected network.

Prototype
dword **CMAPI_Information_GetWLANConnection** (dword deviceID, UTF8* pSSID, dword* pSSIDSize, UTF8* pBSSID, dword* pBSSIDSize)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pSSID	Output	The SSID of the current connection
pSSIDSize	Input/Output	The size of the SSID buffer in bytes.
pBSSID	Output	The BSSID of the current connection
pBSSIDSize	Input/Output	The size of the BSSID buffer in bytes.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000014	Not connected

0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X3000001E	The SSID buffer is not large enough. pSSIDSize contains the minimum required buffer size in bytes.
0X3000001F	The BSSID buffer is not large enough. pBSSIDSize contains the minimum required buffer size in bytes.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.13.13 CMAPI_Information_GetRadioState ()

The **CMAPI_Information_GetRadioState ()** function is used to return the power state of a radio within a device.

Prototype
dword **CMAPI_Information_GetRadioState** (dword deviceID, RadioType Radio, RadioState* pState)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
Radio	Input	Please see the definition of RadioType
pState	Output	Please see the definition of RadioState

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open.
0X00000105	The radio references a radio which the device does not support.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.13.14 CMAPI_Information_GetICCID ()

The **CMAPI_Information_GetICCID ()** function is used to get the ICCID.

Prototype
dword **CMAPI_Information_GetICCID** (dword deviceID, UTF8* pICCID, dword* pICCIDSize)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pICCID	Output	The ICCID value as specified in [ETSI TS 102 221].
pICCIDSize	Input / Output	The size in byte of pICCID buffer.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X30000027	The pICCID buffer is not large enough.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.14 SMS Management APIs

7.14.1 CMAPI_SMS_Send ()

The **CMAPI_SMS_Send ()** function is used to send SMS.

Prototype
dword **CMAPI_SMS_Send** (dword deviceID, dword systemID, SMSRecord* pRecord)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
systemID	Input	The radio system either 3GPP or 3GPP2 to which the function apply when the device is a multi-mode device. • 0x00000000: 3GPP • 0x00000001: 3GPP2

pRecord	Input	The message needs to be sent Note: pPhoneNumber and pMsgContent in the record are available to use to send the message. The message SHALL NOT be stored automatically after sending. If the message is a reply message, the msgID SHALL be the same as the replied one. This case is to address the REPLY PATH issue, see TP-RP in 3GPP TS 23.040. If the message is a new sending one, the msgID SHALL be 0.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000107	System not supported by the device
0X00000130	The device is not in a power state which allows this operation.
0X00005006	The SMS record is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.14.2 CMAPI_SMS_Get ()

The **CMAPI_SMS_Get ()** function is used to retrieve the message.

Prototype
dword **CMAPI_SMS_Get** (dword deviceID, dword systemID, dword msgID, SMSRecord* pRecord, dword* pRecordSize)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
systemID	Input	The radio system either 3GPP or 3GPP2 to which the function apply when the device is a multi-mode device. • 0x00000000: 3GPP • 0x00000001: 3GPP2
msgID	Input	The message ID
pRecord	Output	The SMS record
pRecordSize	Input/Output	The size of the record buffer or if insufficient contains the necessary size

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000107	System not supported by the device
0X00000130	The device is not in a power state which allows this operation.
0X0000500C	The msgID is invalid
0X30005001	The SMS record buffer is not large enough.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.14.3 CMAPI_SMS_Delete ()

The **CMAPI_SMS_Delete ()** function is used to delete SMS.

Prototype
dword **CMAPI_SMS_Delete** (dword deviceID, dword systemID, dword msgID)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
systemID	Input	The radio system either 3GPP or 3GPP2 to which the function apply when the device is a multi-mode device. • 0x00000000: 3GPP • 0x00000001: 3GPP2
msgID	Input	The message ID

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000107	System not supported by the device
0X00000130	The device is not in a power state which allows this operation.
0X0000500C	The msgID is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to

	access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.14.4 CMAPI_SMS_GetIDList ()

The **CMAPI_SMS_GetIDList ()** function is used to get the list of SMS stored on local device or SIM or the terminal device like PC.

Prototype
dword CMAPI_SMS_GetIDList (dword deviceID, dword systemID, dword iFrom, dword* plDList, dword* plDListSize, dword* plDListCount)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
systemID	Input	The radio system either 3GPP or 3GPP2 to which the function apply when the device is a multi-mode device. • 0x00000000: 3GPP • 0x00000001: 3GPP2
iFrom	Input	To indicate where the SMS record is • 0x00000000: from SIM/R-UIM/NAA on UICC • 0x00000001: from local device • 0x00000002: from the terminal device, like PC
plDList	Output	The list of dword values which reference SMS record identifiers.
plDListSize	Input/Output	The size of the ID List buffer in bytes or if insufficient contains the necessary size.
plDListCount	Output	The number of the SMS ID in the list.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000107	System not supported by the device
0X00000130	The device is not in a power state which allows this operation.
0X00005007	The ifrom value is invalid
0X30005004	The size for the plDList buffer is not sufficient, the plDListSize will contain the number of the elements in the list.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.14.5 CMAPI_SMS_Update ()

The **CMAPI_SMS_Update ()** is used to update the status of the SMS.

Note: The msgID in pRecord SHALL be kept even if the SMS content completely changed.

Prototype
dword **CMAPI_SMS_Update** (dword deviceID, dword systemID, SMSRecord* pRecord)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
systemID	Input	The radio system either 3GPP or 3GPP2 to which the function apply when the device is a multi-mode device. • 0x00000000: 3GPP • 0x00000001: 3GPP2
pRecord	Input	The SMS needs to be updated.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000107	System not supported by the device
0X00000130	The device is not in a power state which allows this operation.
0X00005006	The SMS record is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.14.6 CMAPI_SMS_GetSMSCAddress ()

The **CMAPI_SMS_GetSMSCAddress ()** function is used to get the address of SMSC.

Prototype
dword **CMAPI_SMS_GetSMSCAddress** (dword deviceID, dword systemID, UTF8* pSMSCValue, dword* pSMSCValueSize, UTF8* pPSIValue, dword* pPSIValueSize)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
systemID	Input	The radio system either 3GPP or 3GPP2 to which the function apply when the device is a multi-mode device. • 0x00000000: 3GPP • 0x00000001: 3GPP2
pSMSCValue	Output	The address of SMSC.
pSMSCValueSize	Input/Output	The size in byte of pSMSCValue buffer.
pPSIValue	Output	(Optional) The Public Service Identity of the SMSC Note: This field value SHALL be set to NULL if unable to retrieve PSI of SMSC.
PSIValueSize	Input/Output	(Optional) The size in byte of pPSIValue buffer. Note: This field value SHALL be set to 0 if unable to retrieve PSI of SMSC.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000107	System not supported by the device
0X00000130	The device is not in a power state which allows this operation.
0X30005002	SMSCValue buffer is not large enough
0X30005003	PSIValue buffer is not large enough
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.14.7 CMAPI_SMS_SetSMSCAddress ()

The **CMAPI_SMS_SetSMSCAddress ()** function is used to set the address of SMSC.

Prototype
dword **CMAPI_SMS_SetSMSCAddress** (dword deviceID, dword systemID, UTF8* SMSCValue, UTF8* PSIValue)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
systemID	Input	The radio system either 3GPP or 3GPP2 to which the function apply when the device is a multi-mode device. • 0x00000000: 3GPP • 0x00000001: 3GPP2
SMSCValue	Input	The address of the SMSC.
PSIValue	Input	(Optional) The Public Service Identity of the SMSC Note: This field value SHALL be set to NULL if unable to set PSI of SMSC.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000107	System not supported by the device
0X00000130	The device is not in a power state which allows this operation.
0X00005008	The SMSC value is invalid
0X00005009	The PSI value is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.14.8 CMAPI_SMS_GetValidityPeriod ()

The **CMAPI_SMS_GetValidityPeriod ()** function is used to get the validity period setting.

Prototype
dword **CMAPI_SMS_GetValidityPeriod** (dword deviceID, dword* pPeriod)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
pPeriod	Output	The validity period of SMS in minutes

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.14.9 CMAPI_SMS_SetValidityPeriod ()

The **CMAPI_SMS_SetValidityPeriod ()** function is used to set the period of validity of a SMS.

Prototype
dword **CMAPI_SMS_SetValidityPeriod** (dword deviceID, dword Period)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
Period	Input	The duration in minutes the SMSC keeps a message and tries to deliver it.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open

0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.14.10 CMAPI_SMS_GetDeliveryReport ()

The **CMAPI_SMS_GetDeliveryReport ()** function is used to get the delivery report setting, i.e. on or off

Prototype
dword **CMAPI_SMS_GetDeliveryReport** (dword deviceID, dword* pDeliveryReportswitch)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pDeliveryReportswitch	Output	• 0x00000000: switch off • 0x00000001: switch on

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.14.11 CMAPI_SMS_SetDeliveryReport ()

The **CMAPI_SMS_SetDeliveryReport ()** function is used to set the delivery report "On" or "Off".

Prototype
dword **CMAPI_SMS_SetDeliveryReport** (dword deviceID, dword DeliveryReportswitch)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
DeliveryReportswitch	Input	• 0x00000000: switch off • 0x00000001: switch on

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X0000500A	The delivery report switch is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.14.12 CMAPI_SMS_GetRecordCount ()

The **CMAPI_SMS_GetRecordCount ()** function is used to retrieve the number of SMS segments. (Note: one SMS Record equals one or more segments – To obtain the number of SMSRecords, use the function **CMAPI_SMS_GetList ()**).

Prototype
dword **CMAPI_SMS_GetRecordCount** (dword deviceID, dword systemID, dword iFrom, dword* plResult)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
systemID	Input	The radio system either 3GPP or 3GPP2 to which the function apply when the device is a multi-mode device. • 0x00000000: 3GPP • 0x00000001: 3GPP2
iFrom	Input	To indicate where the SMS record is • 0x00000000: from SIM/R-UIM/NAA on UICC • 0x00000001: from local device • 0x00000002: from the terminal device, like PC • Any combination of the above
plResult	Output	The number of SMS segments.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.

0X00000107	System not supported by the device
0X00000130	The device is not in a power state which allows this operation.
0X00005007	The ifrom value is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.14.13 CMAPI_SMS_GetUnreadRecordCount ()

The **CMAPI_SMS_GetUnreadRecordCount ()** function is used to retrieve the number of unread SMS records.

Prototype
dword **CMAPI_SMS_GetUnreadRecordCount** (dword deviceID, dword systemID, dword iFrom, dword* plResult)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
systemID	Input	The radio system either 3GPP or 3GPP2 to which the function apply when the device is a multi-mode device. • 0x00000000: 3GPP • 0x00000001: 3GPP2
iFrom	Input	To indicate where the SMS record is • 0x00000000: from SIM/R-UIM/NAA on UICC • 0x00000001: from local device • 0x00000002: from the terminal device, like PC • Any combination of the above
plResult	Output	The number of the unread SMS record

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000107	System not supported by the device
0X00000130	The device is not in a power state which allows this operation.
0X00005007	The ifrom value is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.15 USSD Management APIs

7.15.1 CMAPI_USSD_Request ()

The **CMAPI_USSD_Request ()** function is used to build up a USSD request to the network.

Prototype
dword **CMAPI_USSD_Request** (dword deviceID, UTF8* USSDData, dword* pUSSDStatus)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
USSDData	Input	The USSD content
pUSSDStatus	Output	The status of the USSD request: • 0x00000000: Done • 0x00000001: Action Required • 0x00000002: Cancelled • 0x00000003: Other client responded • 0x00000004: Network Timeout • 0x00000005: Operation not supported

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.15.2 CMAPI_USSD_Release ()

The **CMAPI_USSD_Release ()** function is used to release the USSD session, if success, the USSD operation will end, without waiting for the release event report from the network.

Prototype
dword **CMAPI_USSD_Release** (dword deviceID, dword* pUSSDStatus)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
pUSSDStatus	Output	The status of the USSD request: • 0x00000000: Done • 0x00000001: Action Required • 0x00000002: Cancelled • 0x00000003: Other client responded • 0x00000004: Network Timeout • 0x00000005: Operation not supported

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.16 GNSS APIs

7.16.1 CMAPI_GNSS_SetState ()

The **CMAPI_GNSS_SetState ()** function is used to set the state of the GNSS functionality on the device.

Prototype
dword **CMAPI_GNSS_SetState** (dword deviceID, dword GNSSState)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
GNSSState	Input	State of GNSS functionality: • 0x00000000: Disabled • 0x00000001: Enabled

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00007001	The GNSS state is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.16.2 CMAPI_GNSS_GetState ()

The **CMAPI_GNSS_GetState ()** function is used to retrieve the state of the GNSS functionality on the device, including whether GNSS is enabled and the state of GNSS tracking.

Prototype
dword **CMAPI_GNSS_GetState** (dword deviceID, dword* pGNSSState, dword* pTrackingStatus)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pGNSSState	Output	State of GNSS: • 0x00000000: Disabled • 0x00000001: Enabled

pTrackingStatus	Output	GNSS tracking status: • 0x00000000: Unknown – cannot be found somewhere else • 0x00000001: Inactive • 0x00000002: Active

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.16.3 CMAPI_GNSS_SetTrackingParameters ()

The **CMAPI_GNSS_SetTrackingParameters ()** function is used to set the values of parameters that control the operation of GNSS tracking on the device.

Prototype
dword **CMAPI_GNSS_SetTrackingParameters** (dword deviceID, dword operation, dword interval, dword timeout, dword accuracy)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
operation	Input	GNSS session operating mode: • 0x00000000: Standalone • 0x00000001: MS Assisted • 0x00000002: MS Based
interval	Input	Interval between position fixes. The value range of interval is in seconds and must be greater than or equal to timeout
timeout	Input	Maximum amount of time used to calculate each position fix (in seconds)
accuracy	Input	Position accuracy threshold (in meters)

Return Values	
Value	**Description**
0X00000000	The function succeeded.

0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00007002	The operation is invalid
0X00007003	The accuracy threshold is not supported
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.16.4 CMAPI_GNSS_GetTrackingParameters ()

The **CMAPI_GNSS_GetTrackingParameters ()** function is used to retrieve the values of parameters that control the operation of GNSS tracking on the device.

Prototype
dword **CMAPI_GNSS_GetTrackingParameters** (dword deviceID, dword* pOperation, dword* pInterval, dword* pTimeout, dword* pAccuracy)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pOperation	Output	GNSS session operating mode: • 0x00000000: Standalone • 0x00000001: MS Assisted • 0x00000002: MS Based
pInterval	Output	Interval between position fixes (in seconds)
pTimeout	Output	Maximum amount of time used to calculate each position fix (in seconds)
pAccuracy	Output	Position accuracy threshold (in meters)

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.16.5 CMAPI_GNSS_SetAGPSConfig ()

The **CMAPI_GNSS_SetAGPSConfig ()** function is used to configure the Assisted GPS (AGPS) server IP address, port number and/or FQDN.

Prototype
dword **CMAPI_GNSS_SetAGPSConfig** (dword deviceID, IPAddress* serverAddress, dword serverPort, UTF8* serverFQDN)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
serverAddress	Input	The IPaddress of AGPS server
serverPort	Input	Port number of AGPS server
serverFQDN	Input	Fully Qualified Domain Name (FQDN) of AGPS server

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00007004	The server address is invalid.
0X00007005	The server port is invalid.
0X00007006	The server FQDN is invalid.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.16.6 CMAPI_GNSS_GetAGPSConfig ()

The **CMAPI_GNSS_GetAGPSConfig ()** function is used to retrieve the values of the Assisted GPS (AGPS) server IP address, port number and FQDN.

Prototype
dword **CMAPI_GNSS_GetAGPSConfig** (dword deviceID, IPAddress* pServerAddress, dword* pServerAddressSize, dword* pServerPort, UTF8* pServerFQDN, dword* pServerFQDNSize)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pServerAddress	Output	Address of AGPS server

pServerAddressSize	Input/Output	The size of the ServerAddress buffer on input. Will contain the minimum byte size needed if input was insufficient.
pServerPort	Output	Port number of AGPS server
pServerFQDN	Output	Fully Qualified Domain Name (FQDN) of AGPS server
pServerFQDNSize	Input/Output	The size of the Server FQDN buffer on input. Will contain the minimum byte size needed if input was insufficient,

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00007011	The ServerAddress buffer needs to be larger, The ServerAddressSize is set to the minimum number of bytes required.
0X00007012	The ServerFQDN buffer needs to be larger. The ServerFQDNSize is set to the minimum number of bytes required.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.16.7 CMAPI_GNSS_SetAutomaticTracking ()

The **CMAPI_GNSS_SetAutomaticTracking ()** function is used to enable and disable automatic GNSS tracking on the device.

Prototype
dword **CMAPI_GNSS_SetAutomaticTracking** (dword deviceID, dword tracking)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
tracking	Input	Automatic tracking session: • 0x00000000: End currently active tracking session • 0x00000001: start an automatic tracking session

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.

0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00007007	The tracking value is invalid
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.16.8 CMAPI_GNSS_GetAutomaticTracking ()

The **CMAPI_GNSS_GetAutomaticTracking ()** function is used to retrieve the state of automatic GNSS tracking on the device.

Prototype
dword **CMAPI_GNSS_GetAutomaticTracking** (dword deviceID, dword* pTracking)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pTracking	Output	State of automatic tracking session: • 0x00000000: No tracking session is active • 0x00000001: A tracking session is active

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.16.9 CMAPI_GNSS_GetDevicePosition ()

The **CMAPI_GNSS_GetDevicePosition ()** function is used to retrieve the current position of the device.

Prototype
dword **CMAPI_GNSS_GetDevicePosition** (dword deviceID, float* pLatitude, float* pLongitude, float* pAltitude, float* pDirection, float* pSpeed, dword *pAccuracy, UTF8* pTimestamp, dword* pTimestampSize)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pLatitude	Output	The current latitude in decimal degrees
pLongitude	Output	The current longitude in decimal degrees
pAltitude	Output	The current altitude in meters
pDirection	Output	The current direction in degrees
pSpeed	Output	The speed in meters per second
pAccuracy	Output	The estimated accuracy of the calculated position in meters
pTimestamp	Output	The Timestamp of the current position. The time format should follow: YYYY-MM-DD HH:MM:SS+HH:MM (-HH:MM). Adheres to ISO 8601.
pTimestampSize	Input/Output	The size of the timestamp buffer or if insufficient contains the necessary size.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X00007013	The timestamp buffer is not large enough.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.16.10 CMAPI_GNSS_SetSystemTime ()

The **CMAPI_GNSS_SetSystemTime ()** function is used to set the value of the system time that will be used by the device's GNSS engine. An accurate system time value directly injected into the GNSS engine can reduce latencies when determining satellite locations as well as the device's actual position.

Prototype
dword **CMAPI_GNSS_SetSystemTime** (dword deviceID, qword systemTime, word numTimeDiscontinuities)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
systemTime	Input	System time value
numTimeDiscontinuities	Input	Number of system time discontinuities

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.17 Data Push Service Management APIs

7.17.1 CMAPI_Push_Enable ()

The **CMAPI_Push_Enable ()** function is used to turn on PUSH option to make applications using the OpenCMAPI Enabler able to receive PUSH messages. This function may be used when the PUSH service is based on different radio types which will be turned on/off individually. If the radio type is set to 0xFF then all PUSH services will be enabled.

Prototype
dword **CMAPI_Push_Enable** (dword deviceID, RadioType Radio)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
Radio	Input	Please see RadioType definition (bitwise combination of one or several types)

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.

0X00000106	The radio references a radio which the device does not support (exception, this error is not reported if the radio is set to 0xFF (all)).
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.17.2 CMAPI_Push_Disable ()

The **CMAPI_Push_Disable ()** function is used to turn off PUSH option to make applications using the OpenCMAPI Enabler unable to receive PUSH messages. This function may be used when the PUSH service is based on different radio types which will be turned on/off individually. If the radio type is set to 0xFF then all PUSH services will be disabled.

Prototype
dword **CMAPI_Push_Disable** (dword deviceID, RadioType Radio)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
Radio	Input	Please see RadioType definition (bitwise combination of one or several types)

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000106	The radio references a radio which the device does not support (exception, this error is not reported if the radio is set to 0xFF (all))
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

7.17.3 CMAPI_Push_GetRadioType ()

The **CMAPI_Push_GetRadioType ()** function is used to get the current bearer type over which the PUSH session is established for an application.

Prototype
dword **CMAPI_Push_GetRadioType** (dword deviceID, RadioType* pRadio)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pRadio	Output	Please see RadioType definition

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

8. CMAPI-2

8.1 Introduction

The CMAPI-2 is an Asynchronous Interface used to provide callbacks (i.e. Notifications) and the registration/deregistration mechanisms to receive these callbacks.

8.2 Registration APIs

This API is exposed by the OpenCMAPI layer.

8.2.1 CMAPI_Callback_Register ()

The **CMAPI_Callback_Register ()** function is used for the application to register for the callbacks which are expected to be received.

Prototype
dword **CMAPI_Callback_Register** (CallbackID ID, callback method)

Parameters		
Field Name	**Mode**	**Description**
ID	Input	See CallbackID definition
method	Input	The callback method to use when event is triggered.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.

8.2.2 CMAPI_Callback_Unregister ()

The **CMAPI_Callback_Unregister ()** function is used to turn off all callbacks or just some.

Prototype
dword **CMAPI_Callback_Unregister** (CallbackID ID)

Parameters		
Field Name	**Mode**	**Description**
ID	Input	See CallbackID definition

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.

8.3 Callback APIs

These callbacks are exposed by the application.

8.3.1 CMAPI_Callback_DetectDevicesComplete ()

The **CMAPI_Callback_DetectDevicesComplete ()** function is used to communicate that a search and validation of the devices in the system is complete. This is a callback method which the OpenCMAPI invokes.

Prototype
dword **CMAPI_Callback_DetectDevicesComplete** (CallbackStatus status, dword devicesPresent, byte* uniqueIdentifierArray)

Parameters		
Field Name	**Mode**	**Description**
status	Input	The status of the callback.
devicesPresent	Input	The number of the devices currently present
uniqueIdentifierArray	Input	An array of 'devicesPresent' strings, each of which uniquely identifies a detected device. The syntax may change from platform to platform, but the unique identifier is guaranteed to be unique to this device on the platform. It MUST not change due to hosting device restart. Example: Windows device GUID. Although this member is declared as a single null-terminated string, it is actually a buffer that can hold multiple null-delimited unique identifiers. Each unique identifier is terminated by a single **NULL** character. The last unique identifier is terminated with a double **NULL** character ("\0\0") to indicate the end of the buffer.

8.3.2 CMAPI_Callback_DeviceChanged ()

The **CMAPI_Callback_DeviceChanged ()** function is used to communicate whenever there is a change in a given device state in particular to indicate that a device has become present or been removed and to notify all applications that have registered for this callback.

This callback could be used to identify the type of device supported for example if the device is used in tethering mode.

Prototype
dword **CMAPI_Callback_DeviceChanged** (dword deviceID, dword devicestate, RadioType radio, dword deviceCapability, dword connectionType, dword deviceType, UTF8* description, UTF8* uniqueIdentifier)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned if the device is already open If the device is not opened: 0
devicestate	Input	The new state of the device. • 0x00000001: Unplugged • 0x00000002: Unavailable • 0x00000003: Available
radio	Input	See RadioType definition
deviceCapability	Input	The additional capabilities not related to radio type supported by the device: • 0x00000000: No additional capability • 0x00000001: GPS • 0x00000002: AGPS in the Control Plane • 0x00000004: AGPS in the User Plane • 0x00000008: Reserved for future use • 0x00000010: Reserved for future use • 0x00000020: Reserved for future use • Any combination of the above
connectionType	Input	The type of the device connection. • 0x00000001: USB • 0x00000002: IRDA • 0x00000004: Bluetooth • 0x00000008: Internal Bus • 0x00000010: Serial • 0x00000020: Wi-Fi • 0x00000040: EmulatedEthernet • Any combination of the above

deviceType	Input	The type of device this message refers to. • 0x00000001: Embedded modem • 0x00000002: USB modem • 0x00000003: Mobile phone acting as modem • 0x00000004: USB modem with Emulated Ethernet • 0x00000005: Wireless Router
description	Input	Description of the device. Intended to be descriptive and displayed by an application.
uniqueIdentifier	Input	The unique identification of this specific device. The syntax may change from platform to platform, but the unique identifier is guaranteed to be unique to this device on the platform. It MUST not change due to hosting device restart. Example: Windows device GUID.

8.3.3 CMAPI_Callback_GetNetworkList_Async_Complete ()

This callback shows the result of a call made to **CMAPI_NetConnectSrv_GetNetworkList_Async ()**.

Prototype
dword **CMAPI_Callback_GetNetworkList_Async_Complete** (CallbackStatus status, dword deviceID, NetworkInfoType* NetworkInfo, dword NetworkInfoCount)

Parameters		
Field Name	**Mode**	**Description**
status	Input	The status of the callback.
deviceID	Input	The ID of the device concerned
NetworkInfo	Input	The Network Information (see NetworkInfoType definition)
NetworkInfoCount	Input	The total number of elements in the array of NetworkInfo

8.3.4 CMAPI_Callback_Connect_Async_Complete ()

The **CMAPI_Callback_Connect_Async_Complete ()** function is invoked as a result of a previous call to **CMAPI_NetConnectSrv_Connect_Async**.

Prototype
dword **CMAPI_Callback_Connect_Async_Complete** (CallbackStatus status, dword deviceID, dword CellularProfileID, dword result)

Parameters		
Field Name	**Mode**	**Description**
status	Input	The status of the callback.
deviceID	Input	The ID of the device concerned
CellularProfileID	Input	The ID of the Cellular Profile it applies to.
result	Input	• 0x00000000: The connection succeeded • 0x00000001: The connect attempt failed, reason: The network

		connection was refused by network • 0x00000002: The connect attempt failed, reason: TBD

8.3.5 CMAPI_Callback_Disconnect_Async_Complete ()

The **CMAPI_Callback_Disconnect_Async_Complete ()** function is invoked as a result of a previous call to **CMAPI_NetConnectSrv_Disconnect**.

Prototype
dword **CMAPI_Callback_Disconnect_Async_Complete** (CallbackStatus status, dword deviceID, dword CellularProfileID, dword result)

Parameters		
Field Name	**Mode**	**Description**
status	Input	The status of the callback.
deviceID	Input	The ID of the device concerned
CellularProfileID	Input	The ID of the Cellular Profile it applies to
result	Input	• 0x00000000: The disconnect operation succeeded • 0x00000001: The disconnect attempt failed, reason: TBD

8.3.6 CMAPI_Callback_CancelConnect_Async_Complete ()

The **CMAPI_Callback_CancelConnect_Async_Complete ()** function is invoked as a result of a previous call to **CMAPI_NetConnectSrv_CancelConnect_Async**.

Prototype
dword **CMAPI_Callback_CancelConnect_Async_Complete** (CallbackStatus status, dword deviceID, dword CellularProfileID, dword result)

Parameters		
Field Name	**Mode**	**Description**
status	Input	The status of the callback.
deviceID	Input	The ID of the device concerned
CellularProfileID	Input	The ID of the Cellular Profile it applies to
result	Input	• 0x00000000: The connect operation was cancelled. • 0x00000001: The cancel operation failed, reason: TBD

8.3.7 CMAPI_Callback_SessionStateChange ()

The **CMAPI_Callback_SessionStateChange ()** function is used to communicate the session state change

Prototype
dword **CMAPI_Callback_SessionStateChange** (dword deviceID, dword CellularProfileID, dword connectionStatus)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
CellularProfileID	Input	The ID of the Cellular Profile it applies to
connectionStatus	Input	The new status of the connection of the device: • 0x00000000: Connected • 0x00000001: Disconnected (it may be possible to distinguish between passive and active disconnection) • 0x00000002: Connecting • 0x00000003: Disconnecting • 0x00000004: Scanning • 0x00000010: Unknown state

8.3.8 CMAPI_Callback_BearerStatusChange ()

The **CMAPI_Callback_BearerStatusChange ()** function is used to communicate a bearer status change

Prototype
dword **CMAPI_Callback_BearerStatusChange** (dword deviceID, dword bearer)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
bearer	Input	Indication of the bearer: • 0x00000001: No Service • 0x00000002: Any packet oriented service • 0x00000004: Any circuit switched service • 0x00000010: GSM service • 0x00000020: GPRS service • 0x00000040: EDGE service • 0x00000080: CDMA service • 0x00000100: QNC service • 0x00000200: 1X-RTT service

		• 0x00000400: EV-DO service • 0x00000800: EV-DV service • 0x00001000: IOTA service • 0x00002000: IOTA REVA service • 0x00004000: UMTS service • 0x00008000: HSDPA service (Included for legacy purpose, not all operators use HSDPA+) • 0x00010000: HSUPA service • 0x00020000: HSPA Plus service • 0x00040000: PHS service • 0x00080000: FOMA service • 0x00100000: LTE service • 0x10000000: WLAN service

8.3.9 CMAPI_Callback_TrafficChannelDormancy ()

The **CMAPI_Callback_TrafficChannelDormancy ()** function is used to communicate the changes in the traffic level.

Prototype
dword **CMAPI_Callback_TrafficChannelDormancy** (dword deviceID, dword state)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
state	Input	The new traffic channel dormancy state • 0x00000000: Dormant. See definitions section. Marked Dormant after 10 seconds of no use. • 0x00000001: Traffic channel in use.

8.3.10 CMAPI_Callback_CDMA2000ActivationState ()

The **CMAPI_Callback_CDMA2000ActivationState ()** function is used to communicate the changes in the CDMA 2000 Activation state

Prototype
dword **CMAPI_Callback_CDMA2000ActivationState** (CallbackStatus status, dword deviceID, dword state)

Parameters		
Field Name	**Mode**	**Description**
status	Input	The status of the callback.
deviceID	Input	The ID of the device concerned
state	Input	The new activation state

		• 0x00000000: Service not activated • 0x00000001: Service activated • 0x00000002: Activation connecting • 0x00000003: Activation connected • 0x00000004: OTASP security authenticated • 0x00000005: OTASP NAM downloaded • 0x00000006: OTASP MDN downloaded • 0x00000007: OTASP IMSI downloaded • 0x00000008: OTASP PRL downloaded • 0x00000009: OTASP SPC downloaded • 0x0000000A: OTASP settings committed.

8.3.11 CMAPI_Callback_SearchWLANNetworkComplete ()

The **CMAPI_Callback_SearchWLANNetworkComplete ()** function is called when a search for a particular WLAN network has been completed. The function is invoked as a result of a previous call to **CMAPI_WLAN_SearchNetwork_Async ()**.

Prototype
dword **CMAPI_Callback_SearchWLANNetworkComplete** (CallbackStatus status, dword deviceID, WLANNetwork* pNetwork, dword Present)

Parameters		
Field Name	**Mode**	**Description**
status	Input	The status of the callback.
deviceID	Input	The ID of the device concerned
pNetwork	Input	The network identification.
Present	Input	The presence status of the Wlan network searched • 0x00000000: Not present • 0x00000001: Present

8.3.12 CMAPI_Callback_RadioState ()

The **CMAPI_Callback_RadioState ()** function is used to communicate changes in the radio power state.

Prototype
dword **CMAPI_Callback_RadioState** (dword deviceID, RadioType radio, RadioState state)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The device whose radio has changed power state.
radio	Input	Please see RadioType definition
state	Input	Please see RadioState definition

8.3.13 CMAPI_Callback_SetRadioState_Async_Complete ()

The **CMAPI_Callback_SetRadioState_Async_Complete ()** function is invoked as a result of a previous call to **CMAPI_DevSrv_SetRadioState_Async**.

Prototype
dword **CMAPI_Callback_SetRadioState_Async_Complete** (CallbackStatus status, dword deviceID, dword result)

Parameters		
Field Name	**Mode**	**Description**
status	Input	The status of the callback.
deviceID	Input	The ID of the device concerned
result	Input	• 0x00000000: The change of the power state succeeded • 0x00000001: The change of the power state failed, reason: The radio references a radio which the device does not support. • 0x00000002: The connect attempt failed, reason: The device does not support the indicated power state. (ex power saving)

8.3.14 CMAPI_Callback_Roaming ()

The **CMAPI_Callback_Roaming ()** function is used indicate changes in Roaming status

Prototype
dword **CMAPI_Callback_Roaming** (dword deviceID, dword state)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
state	Input	The Indication of the roaming state • 0x00000000: Home (not roaming) • 0x00000001: Roaming

8.3.15 CMAPI_Callback_SignalStrength ()

The **CMAPI_Callback_SignalStrength ()** function is used to return the current signal strength value, the percentage of signal present and the signal quality.

Prototype
dword **CMAPI_Callback_SignalStrength** (dword deviceID, RadioType radio, dword SignalStrengthRaw, dword SignalStrengthPercent, dword SignalQualityPercent)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
radio	Input	Please see RadioType definition
SignalStrengthRaw	Input	The signal strength value in dBm.
SignalStrengthPercent	Input	The signal strength as a percentage - SHOULD be adjusted to device capabilities.
SignalQualityPercent	Input	The signal quality as a percentage - SHOULD be adjusted to device capabilities.

8.3.16 CMAPI_Callback_GNSS ()

The **CMAPI_Callback_GNSS ()** function is used to indicate a change in the GNSS state.

Prototype
dword **CMAPI_Callback_GNSS** (dword deviceID, dword state, dword fix, float latitude, float longitude, float altitude, float direction, float speed, dword accuracy, UTF8* timestamp)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
state	Input	Indication of the GNSS state • 0x00000000: GNSS off • 0x00000001: GNSS on
fix	Input	Indication if the GNSS has a fix • 0x00000000: No fix • 0x00000001: Fix
latitude	Input	The current latitude in decimal degrees
longitude	Input	The current longitude in decimal degrees
altitude	Input	The current altitude in meters
direction	Input	The current direction in degrees
speed	Input	The speed in meters per second
accuracy	Input	The estimated accuracy of the current position in meters
timestamp	Input	The Timestamp of the current position. The time format should follow: YYYY-MM-DD HH:MM:SS+HH:MM (-HH:MM). Adheres to ISO 8601.

8.3.17 CMAPI_Callback_SMS ()

The **CMAPI_Callback_SMS ()** function is used to indicate that a new SMS message has been received and the number of segments in the mailbox.

Note: The connection manager MAY select either **CMAPI_Callback_SMS ()** or **CMAPI_Callback_SMS_Message ()** to retrieve the notification of a new SMS message. It is recommended not to use both together to prevent the Connection Manager Application from being alerted twice for one notification.

Prototype
dword **CMAPI_Callback_SMS** (dword deviceID, dword systemID, dword msgID, dword mailbox, dword totalSegments, dword newSegments)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
systemID	Input	The radio system either 3GPP or 3GPP2 to which the function apply when the device is a multi-mode device. • 0x00000000: 3GPP • 0x00000001: 3GPP2
msgID	Input	The message ID
mailbox	Input	Indication of the mailbox • 0x00000000: in the SIM/R-UIM/NAA on UICC; • 0x00000001: in the local device; • 0x00000002: in the terminal device, like PC 0xFFFFFFFF: None, for display directly message only.
totalSegments	Input	The total number of segments in the mailbox. The number corresponds to the value returned by the function **CMAPI_SMS_GetRecordCount()**
newSegments	Input	The current number of new segments in the mailbox

8.3.18 CMAPI_Callback_SMS_Message ()

The **CMAPI_Callback_SMS_Message ()** function is used to provide to application the new received message while not only a notice that a new message is received.

Note: For concatenated SMS, the callback will be invoked every time a segment or package arrives. For example, if there are two segments or packages for one SMS, then the callback will be invoked twice. The Connection Manager Application could determine the completeness of retrieval by checking whether the totalPack and currentPack are the same or not.

Prototype
dword **CMAPI_Callback_SMS_Message** (dword deviceID, dword systemID, SMSRecord* pRecord)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
systemID	Input	The radio system either 3GPP or 3GPP2 to which the function apply when the device is a multi-mode device. • 0x00000000: 3GPP • 0x00000001: 3GPP2
pRecord	Input	The SMS record

8.3.19 CMAPI_Callback_ByteCount

The **CMAPI_Callback_ByteCount ()** function is used to indicate the current byte count. This is a periodic notification. This callback SHALL be made immediately when the application registers for this message. The callback SHALL also occur at a maximum of every 15 seconds when the connection is not Dormant. The OpenCMAPI implementation is free to make this callback sooner if deemed useful, in any event the callback MAY NOT occur with greater frequency than once a second. The byte count accumulates between the last connection and either a manual disconnect or some other event that causes the radio to be in disconnected state. This callback must not occur while in the disconnected state.

Prototype
dword **CMAPI_Callback_ByteCount** (dword deviceID, dword CellularProfileID, qword Tx, qword Rx, dword wrapped)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
CellularProfileID	Input	The ID of the Cellular Profile it applies to
Tx	Input	The current count of Tx bytes.
Rx	Input	The current count of Rx bytes
wrapped	Input	This is used to denote when Tx and/or Rx counters have overflowed. Counting will continue like normal and the indication will be set once for each overflow. The following definition is a bitwise combination and allows for Tx and/or Rx to be set at the same time. • 0x00000000: No Overflow • 0x00000001: Tx overflow

		• 0x00000002: Rx overflow

8.3.20 CMAPI_Callback_USSD ()

The **CMAPI_Callback_USSD ()** function is used to communicate a USSD message.

Prototype
dword **CMAPI_Callback_USSD** (dword deviceID, dword status, UTF8* data)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
status	Input	The status • 0x00000000: Done • 0x00000001: Action Required • 0x00000002: Cancelled • 0x00000003: Other client responded • 0x00000004: Network Timeout
data	Input	The contents of the message

8.3.21 CMAPI_Callback_QoSChange ()

The **CMAPI_Callback_QoSChange ()** function is used to communicate a change in QoS as defined in [3GPP TS 23.107].

Prototype
dword **CMAPI_Callback_QoSChange** (dword deviceID, dword CellularProfileID, QoSStructure* QoS)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
CellularProfileID	Input	The ID of the Cellular Profile it applies to
QoS	Input	See QoS Structure definition

8.3.22 CMAPI_Callback_RFInformationChange ()

The **CMAPI_Callback_RFInformationChange ()** function is used to communicate a change related to RF.

Prototype
dword **CMAPI_Callback_RFInformationChange** (dword deviceID, UTF8* radioTechnology, UTF8* bandClass, UTF8* channel)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
radioTechnology	Input	Name of the technology in use
bandClass	Input	Name of the band class in use
channel	Input	Name of the channel in use

8.3.23 CMAPI_Callback_PINPUKStatus ()

The **CMAPI_Callback_PINPUKStatus ()** function is used to return the status of the PINs/PUKs for all active NAAs as soon as the status changes by any OpenCMAPI applications or any other applications.

Prototype
qword **CMAPI_Callback_PINPUKStatus** (dword deviceID, PINPUKStatusType* PINPUKStatusList, dword PINPUKStatusListCount)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
PINPUKStatusList	Input	The status of the PINs/PUKs for all active NAAs (see PINPUKStatusType definition)
PINPUKStatusListCount	Input	The number of entries in the status list

8.3.24 CMAPI_Callback_ScanWLANComplete ()

The **CMAPI_Callback_ScanWLANComplete ()** function is used to notify that a scan for WLAN networks has been completed. The function is invoked as a result of a previous call to **CMAPI_WLAN_Scan_Async ()**.

Prototype
dword **CMAPI_Callback_ScanWLANComplete** (CallbackStatus status, dword deviceID, dword networks)

Parameters		
Field Name	**Mode**	**Description**
status	Input	The status of the callback.
deviceID	Input	The ID of the device concerned
networks	Input	The number of networks in the current scan list.

8.3.25 CMAPI_Callback_WLANNewAvailableNetwork ()

The **CMAPI_Callback_WLANNewAvailableNetwork ()** function is used to notify that a new network has been discovered. It is recommended to use **CMAPI_WLAN_GetScanResults** to get network list if the awCount has not just increased of 1 increment compared to previous result.

Prototype
dword **CMAPI_Callback_WLANNewAvailableNetwork** (dword deviceID, dword awCount, Located_WLANNetwork* pNetwork)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
awCount	Input	The number of WLAN networks available.
pNetwork	Input	The new network which has been located. Please see Located_WLANNetwork

8.3.26 CMAPI_Callback_WLANConnectionStatus ()

The **CMAPI_Callback_WLANNotification ()** function is used to receive WLAN connection Status.

Prototype
dword **CMAPI_Callback_WLANConnectionStatus** (CallbackStatus status, dword deviceID, dword connectionstatus)

Parameters		
Field Name	**Mode**	**Description**
status	Input	The status of the callback.
deviceID	Input	The ID of the device concerned
connectionstatus	Input	WLAN event: • 0x00000000: Connection attempt starting • 0x00000001: Attempting association • 0x00000002: Association failed • 0x00000003: Attempting authentication • 0x00000004: Authentication failed • 0x00000005: Requesting IP address • 0x00000006: IP grant failed • 0x00000010: Connected • 0x00000020: Disconnecting • 0x00000021: Disconnected

8.3.27 CMAPI_Callback_PUSHReceived ()

The **CMAPI_Callback_PUSHReceived ()** function is used to notify an application when a new PUSH message has been received.

Prototype
dword **CMAPI_Callback_PUSHReceived** (dword deviceID, UTF8* contentType, UTF8* applicationID, byte* data, dword length)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The device concerned
contentType	Input	The content type carried in the PUSH message
applicationID	Input	The application id carried in the PUSH message (application ID in this context is the ID of the PUSH application)
data	Input	The contents of the PUSH message in binary form.
length	Input	The length of the data in bytes.

8.3.28 CMAPI_Callback_OMADMStatus ()

This callback indicates any OMA-DM operation Progress or Status inbetween. The User Action flag will be set if Device API requires user action to be applied.

Prototype
dword **CMAPI_Callback_OMADMStatus** (CallbackStatus status, dword deviceID, dword OMADMOperation, dword OMADMStatus, boolean userActionRequired)

Parameters		
Field Name	**Mode**	**Description**
status	Input	The status of the callback.
deviceID	Input	The ID of the device concerned
OMADMOperation	Input	Device management operation under consideration. • 0x00000001: HFA (Hands Free Activation) • 0x00000002: HFA_CIDC (CIDC during HFA process) • 0x00000003: HFA_CIPRL (CIPRL during HFA process) • 0x00000004: HFA_CIFUMO (CIFUMO during HFA process) • 0x00000005: CIDC • 0x00000006: CIPRL • 0x00000007: CIFUMO • 0x00000008: NIDC • 0x00000009: NIPRL • 0x0000000A: NIFUMO

OMADMStatus	Input	The status of all OMA-DM Operation listed below shall be informed via the OMA-DM callback function along with the above OMADMOperation. • 0x00000001: OMADM_STARTED (indicates the particular OMA-DM Session is Started) • 0x00000002: OMADM_PROGRESS • 0x00000003: OMADM_RESULT _SUCCESS (indicates the particular OMA-DM Session is Completed succesfully) • 0x00000004: OMADM_RESULT_FAILURE (indicates the particular OMA-DM Session Failed) • 0x00000005: OMADM_RESULT_CANCELED (result in case of user cancelling OMA-DM Session) • 0x00000006: OMADM_SESSION_PROGRESS (particular OMA-DM Session is in progress) • 0x00000007: OMADM_PACKAGE_AVAILABLE (FUMO package available) • 0x00000008: OMADM_PACKAGE_DOWNLOADED (FUMO package downloaded) • 0x00000009: OMADM_UPDATE_NOT_AVAILABLE (PRL update or FUMO update Not available) • 0x0000000A: OMADM_NOTIFICATION_SENT_TO_SERVER (Device OMA-DM client sending final status notification to OMA Server completed) • 0x0000000B: OMADM_HFA_START • 0x0000000C: OMADM_HFA_CIFUMO • 0x0000000D: OMADM_HFA_CIPRL • 0x0000000E: OMADM_HFA_CIDC • 0x0000000F: OMADM_HFA_END
userActionRequired	Input	Indicates whether user action is required in response to the status received. Usually applies to FUMO Package Download and Update Device with FUMO Package. • 0: No User Action Required • 1: User Action Required

8.3.29 CMAPI_Callback_UICC_ToolKitProactiveCommand ()

The device SHALL support the class s, “Support of CAT over the modem interface”, as specified in [ETSI TS 102 223].

The **CMAPI_Callback_UICC_ToolKitProactiveCommand ()** function is used to receive the ToolKit Proactive Commands sent by the SIM/R-UIM/UICC and routed to the Connection Manager Application by the device (see [ETSI TS 102 223] for the routing aspects).

The device SHALL send this callback only when the Connection Manager Application support the corresponding ToolKit Proactive Commands as previously indicated into the **CMAPI_UICC_SetTerminalProfile ()** and when no overlap was detected into the **CMAPI_UICC_SetTerminalProfile ()**.

The device SHALL send this callback as soon as it receives the ToolKit Proactive Commands from the SIM/R-UIM/UICC.

Prototype
dword **CMAPI_Callback_UICC_ToolKitProactiveCommand** (dword deviceID, byte toolKitProactiveCommand[256])

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
toolKitProactiveCommand	Input	ToolKit Proactive Command in hexadecimal format as specified in [ETSI TS 102 223] for the core part, in [3GPP TS 31.111] for the 3GPP specific part, in [C.S0035] for the 3GPP2 specific part.

8.3.30 CMAPI_Callback_UICC_DeviceTerminalProfile ()

The device SHALL support the class s, “Support of CAT over the modem interface”, as specified in [ETSI TS 102 223].

The **CMAPI_Callback_UICC_DeviceTerminalProfile ()** function is used for the Connection Manager Application to receive the TERMINAL PROFILE sent by the device to the SIM/R-UIM/UICC each time the device sent it.

The device SHALL send this callback at the same time it sends the TERMINAL PROFILE to the SIM/R-UIM/UICC.

Prototype
dword **CMAPI_Callback_UICC_DeviceTerminalProfile** (dword deviceID, byte terminalProfile[256])

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
terminalProfile	Input	The hexadecimal value of the TERMINAL PROFILE as specified in the chapter “Structure and coding of the TERMINAL PROFILE” of [ETSI TS 102 223] for the core part, in the chapter “Structure and coding of the TERMINAL PROFILE” of [3GPP TS 31.111] for the 3GPP specific part, in the chapter “Structure and coding of the TERMINAL PROFILE” of [3GPP2 C.S0035] for the 3GPP2 specific part.

8.3.31 CMAPI_Callback_VerifyPIN ()

The **CMAPI_Callback_VerifyPIN ()** function is used to signal that a PIN should be collected from the user and supplied to the API through the **CMAPI_DevSrv_VerifyPIN** method.

Prototype
dword **CMAPI_Callback_VerifyPin** (dword deviceID)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The device for which the PIN is needed.

8.3.32 CMAPI_Callback_PermittedBearersChange ()

The **CMAPI_Callback_PermittedBearersChange ()** function is used to notify that a change occurred in the PermittedBearers for the device.

Prototype
dword **CMAPI_Callback_PermittedBearersChange** (dword deviceID, dword bearers)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
bearers	Input	Indication of bearer(s) permitted (bitmap): • 0x00000001: GSM • 0x00000002: WCDMA/UMTS • 0x00000004: CDMA • 0x00000008: EVDO • 0x00000010: TD_SCDMA • 0x00000020: LTE

8.3.33 CMAPI_Callback_NetConnectSrv_SecondaryPDPContext_Connect_Async_Complete ()

The **CMAPI_Callback_NetConnectSrv_SecondaryPDPContext_Connect_Async_Complete ()** function is invoked as a result of a previous call to **CMAPI_NetConnectSrv_SecondaryPDPContext_Connect_Async**.

Prototype
dword **CMAPI_Callback_NetConnectSrv_SecondaryPDPContext_Connect_Async_Complete** (CallbackStatus status, dword deviceID, dword CellularProfileID, byte SecondaryContextnumber, dword result)

Parameters		
Field Name	**Mode**	**Description**
status	Input	The status of the callback.
deviceID	Input	The ID of the device concerned
CellularProfileID	Input	The ID of the Cellular Profile it applies to
SecondaryContext number	Input	Secondary context number from 1 to 16.
result	Input	• 0x00000000: The connection succeeded • 0x00000001: The connect attempt failed, reason: The network connection was refused by network • 0x00000002: The connect attempt failed, reason: TBD

8.3.34 CMAPI_Callback_NetConnectSrv_SecondaryPDPContext_Disconnect_Async_Complete ()

The **CMAPI_Callback_NetConnectSrv_SecondaryPDPContext_Disconnect_Async_Complete ()** function is invoked as a result of a previous call to **CMAPI_NetConnectSrv_SecondaryPDPContext_Disconnect_Async**.

Prototype
dword **CMAPI_Callback_NetConnectSrv_SecondaryPDPContext_Disconnect_Async_Complete** (CallbackStatus status, dword deviceID, dword CellularProfileID, byte SecondaryContextnumber, dword result)

Parameters		
Field Name	**Mode**	**Description**
status	Input	The status of the callback.
deviceID	Input	The ID of the device concerned
CellularProfileID	Input	The ID of the Cellular Profile it applies to
SecondaryContext number	Input	Secondary context number from 1 to 16.
result	Input	• 0x00000000: The disconnect operation succeeded • 0x00000001: The disconnect attempt failed, reason: TBD

8.3.35 CMAPI_Callback_NetConnectSrv_SecondaryPDPContext_CancelConnect_Async_Complete ()

The **CMAPI_Callback_NetConnectSrv_SecondaryPDPContext_CancelConnect_Async_Complete ()** function is invoked as a result of a previous call to **CMAPI_NetConnectSrv_SecondaryPDPContext_CancelConnect_Async**.

Prototype
dword **CMAPI_Callback_NetConnectSrv_SecondaryPDPContext_CancelConnect_Async_Complete** (CallbackStatus status, dword deviceID, dword CellularProfileID, byte SecondaryContextnumber, dword result)

Parameters		
Field Name	**Mode**	**Description**
status	Input	The status of the callback.
deviceID	Input	The ID of the device concerned
CellularProfileID	Input	The ID of the Cellular Profile it applies to
SecondaryContext number	Input	Secondary context number from 1 to 16.
result	Input	• 0x00000000: The connect operation was cancelled. • 0x00000001: The cancel operation failed, reason: TBD

9. Return Values & Error Codes

9.1 Return Values and Error Codes

The Return values and Error Codes table is used to capture the warnings, error codes and information when the Open CMAPI is running. Some additional warnings and output information can be defined depending on the implementation.

Return Values & Error Codes	
Value	**Description**
General Return Values & Error Codes	
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000002	Invalid Parameter
0X00000003	Buffer size not large enough
0X00000004	Invalid Operation
0X00000005	No service
0X00000006	The requested operation cannot currently be completed because another application is currently performing the same operation.
0X00000007	This optional function is not supported by this implementation
0X00000010	The OpenCMAPI implementation cannot perform this operation since there is currently a connection which prevents the request. NOTE: The OpenCMAPI implementation may be able to apply the change in some conditions and may return success instead of this return code in some connected conditions.
0X00000011	The type of data requested is not present
0X00000013	QoS unsupported
0X00000014	Not connected
Device Error Codes	
0X00000100	The UniqueIdentifier is referencing a non-existing device
0X00000101	The deviceID references a non-existing device or a device which is not open
0X00000102	The device is already opened.
0X00000103	Maximum number of device that the API can handle per client is reached (can be 1), close another open device handle.
0X00000104	The device does not contain hardware which supports this operation.
0X00000105	The radio references a radio which the device does not support
0X00000106	The radio references a radio which the device does not support (exception, this error is not reported if the radio is set to 0xFF (all)).
0X00000107	System not supported by the device
0X00000108	The requested data is not meaningful for a 3GPP device.
0X00000109	The requested data is not meaningful for a 3GPP2 device.
0X00000110	The device cannot be activated while connected.

0X00000111	The device is not connected
0X00000120	Configuration not supported by the device
0X00000121	The device does not offer this capability
0X00000130	The device is not in a power state which allows this operation.
0X00000131	Requested power state is not supported by the device (ex power saving)
0X00000132	Radio off
0X00000133	The power state is invalid
0X00000134	The system ID is invalid
0X00000135	No IMSI available
0X00000210	Control Key not supported by this system (when an ID of a 3GPP2 only Control Key is sent to a 3GPP system device or when an ID of a 3GPP only Control Key is sent to a 3GPP2 system device).
0X00000211	The control key value is invalid
UICC Error Codes	
0X00000501	There is no smart card support for this device
0X00000502	Smart card not accessible
0X00000551	ENVELOPE command was not sent to SIM/R-UIM/UICC as overlapping was detected.
0X00000552	The envelope command is invalid
0X00000553	The terminal profile is invalid
0X00000554	The function succeeded except for the overlapping ToolKit functions with the device or another or other Connection Manager Application(s)
0X00000555	The terminal response is invalid
Profile Error Codes	
0X00002001	The Cellular profile ID does not exist
0X00002002	The Cellular profile ID is not valid
0X00002003	The Cellular profile ID already exists, this only happen when creating a profile with an existing ID
0X00002004	The Cellular profile can not be updated while currently in use (connected)
0X00002005	A default profile has not been set for this device.
0X00002101	The user name is not valid
0X00002102	The password is not valid
0X00002104	The APN is not valid
0X00002105	The IP Address is not valid
0X00002106	The primary DNS address is not valid
0X00002107	The secondary DNS address is not valid
0X00002108	The Auth type is not valid
0X00002109	The IPAddrType is not valid

0X0000210A	The profile type is not valid
0X0000210B	The timeout is not valid
0X00002202	The type of IP address is not available.
Network Connection Error Codes	
0X00003001	The requested bearer is not possible
0X00003002	There is no connection to disconnect from
0X00003004	There is no connecting session for cancellation
0X00003005	The Connection is releasing
0X00003006	Remote system not present
0X00003007	The supplied index identifies a record which does not exist.
0X00003008	Current APN cannot be retrieved because there is no connection.
0X00003009	The requested connection type is not valid
0X0000300A	There is currently a connection which prevents this operation. It is necessary to disconnect before the requested operation can be completed.
0X00003101	The requested mode is not valid
0X00003102	The requested PLMNID is not valid
0X00003103	The requested bearer or combination of bearers is not valid.
0X00003201	No Primary context activated
0X00003202	The secondary context doesn't exist
0X00003203	The secondary context is already activated/created
0X00003204	The secondary context activation is in progress
0X00003205	The secondary context is already deactivated
0X00003206	The secondary context deactivation is in progress
0X00003207	The secondary context is already deactivating
CDMA 2000 Error Codes	
0X00004001	Unrecognized session identifier.
0X00004002	The SPC is valid.
0X00004003	The SPC is invalid.
0X00004004	The requested activation code is invalid.
0X00004005	Activation failed (other than invalid activation code).
0X00004006	The index is invalid
0X00004007	File does not exist at the given path.
0X00004008	An invalid PRL file is entered.
0X0000400B	No record exists at the specified index.
0X0000400C	The ACCOLC is invalid.
0X0000400D	The requested ForceRev0 is invalid

0X0000400E	The CustomSCP is invalid
0X0000400F	The protocol is invalid
0X00004010	The broadcast is invalid
0X00004011	The application is invalid
0X00004012	The roaming is invalid
0X00004013	The SID is invalid
0X00004014	The MDN is invalid
0X00004015	The MIN is invalid
0X00004016	The PRL is invalid
0X00004017	The MNHA is invalid
0X00004018	The MNAAA is invalid
0X00004019	The session type is invalid
0X0000401A	The session state is invalid
0X0000401B	The failure reason is invalid
0X0000401C	The retry count is invalid
0X0000401D	The session pause is invalid
0X0000401E	The selection is invalid
0X0000401F	The session id is invalid
0X00004020	The defer is invalid
0X00004021	The feature state is invalid
0X00004022	The update feature state is invalid.
0X00004023	The firmware update feature state is invalid
0X00004024	The reason is invalid
0X00004025	The mode is invalid
0X00004026	The enabled value is invalid
0X00004027	The RevTunn value is invalid
0X00004028	The NAI is invalid
0X00004029	The HASPI is invalid
0X0000402A	The AAASPI is invalid
0X0000402B	The Address parameter was not formatted properly.
0X0000402C	The Primary Home Agent parameter was not formatted properly.
0X0000402D	The Secondary Home Agent parameter was not formatted properly.
0X0000402E	The retry limit is invalid
0X0000402F	The retry interval is invalid
0X00004030	The Reregperiod is invalid
0X00004031	The Reregtraffic is invalid

0X00004032	The HAAuthenticator is invalid
0X00004033	The HA2002bis is invalid
SMS Error Codes	
0X00005001	Failure of communication with device
0X00005002	Timer expired without receiving response from device
0X00005003	Response with error indication from device
0X00005004	Operation NOT supported
0X00005005	SMS message NOT found
0X00005006	The SMS record is invalid
0X00005007	The ifrom value is invalid
0X00005008	The SMSC value is invalid
0X00005009	The PSI value is invalid
0X0000500A	The delivery report switch is invalid
0X0000500B	The SMS Class is invalid
0X0000500C	The msgID is invalid
0X00005901	The USSD Data is invalid
Information Status Error Codes	
0X00006001	The type of data requested is not present
0X00006002	The type is not valid
0X00006003	Remote system not present
0X00006004	The supplied index identifies a record which does not exist.
0X00006005	Current APN cannot be retrieved because there is no connection.
0X00006006	The type of IP address is not available.
0X00006007	IP Address is not currently assigned (advisable to retry call)
0X00006008	Authentication failure
GNSS Error Codes	
0X00007001	The GNSS state is invalid
0X00007002	The operation is invalid
0X00007003	The accuracy threshold is not supported
0X00007004	The server address is invalid.
0X00007005	The server port is invalid.
0X00007006	The server FQDN is invalid.
0X00007007	The tracking value is invalid
WLAN Error Codes	
0X00010001	No network exists at the specified index.
0X00010002	Predefined networks are not able to be modified.

0X00010004	The SSID is invalid
0X00010005	The BSSID is invalid
0X00010006	The Friendly Name is invalid
0X00010007	The security parameter is invalid
0X00010008	The mode parameter is invalid
0X00010009	The hidden parameter is invalid
0X0001000A	The key is invalid
0X0001000B	The EAP authentication method is invalid
0X0001000C	The EAP configuration is invalid
0X0001000D	The Wlan Encryption Type is invalid
0X00011001	There is no existing WLAN connection
0X00011002	Security mode does not allow connectivity to unknown networks.
0X00011005	Operation is prohibited by security policy.
0X00011006	No pending operation.
0X00011007	The pin for WPS was malformed or incorrect size
0X00011008	The device is not connected
0X00011009	Device (i.e.: WLAN only device that does not support NAA on UICC for authentication) does not support the requested function.
0X00012001	The SSID does not reference a valid known network.
0X00012002	The BSSID does not reference a valid known network
0X00012003	IP Address is not currently assigned (advisable to retry call)
0X00012004	Authentication failure
0X00013001	Invalid combination of AUTH and CIPHER
0X00013002	Index NOT referring to a valid known network
0X00013003	NO existing WLAN connection
0X00013004	IP address NOT valid
0X00013005	Subnet mask NOT valid
0X00013006	Operation prohibited by security policy
0X00013007	The specified index is to large and would leave a gap in the known networks list
0X00013008	Index is not valid for user defined networks. Please try a higher index.
0X00013009	The mode is invalid
0X0001300A	The address is invalid
0X0001300B	The subnet mask is invalid
0X0001300C	The http proxy is invalid
0X0001300D	The mac address is invalid
0X0001300E	The default gateway is invalid

PIN/PUK management Error Codes	
	SW1 and SW2 are the Status Words provided by the SIM/R-UIM/UICC (see next chapter). If no Status Word is provided, SW1SW2 will be replaced by “0000”.
0X1001SW1SW2	Wrong PIN.
0X1002SW1SW2	PIN is blocked. PUK (UNBLOCK PIN) needed.
0X1003SW1SW2	Wrong Old PIN.
0X1004SW1SW2	Old PIN is blocked. PUK (UNBLOCK PIN) needed.
0X1005SW1SW2	Wrong PUK.
0X1006SW1SW2	PUK (UNBLOCK PIN) blocked.
0X1007SW1SW2	Invalid parameter(s)
0X11000001	The NAA Name is invalid
0X11000002	The PIN Type is invalid
0X11000003	The PUK Type is invalid
Buffer Error Codes	
	Listing all buffer error codes
0X30000000	The OpenCMAPIVersion buffer is not large enough
0X30000001	The buffer is not sufficient to hold the data, pCellularProfileSize will contain the minimum number of bytes required.
0X30000002	The buffer is not sufficient to hold the data, the pCellularProfileIDListSize will contain the minimum number of bytes required.
0X30000003	The size of the network info buffer is insufficient. pNetworkInfoSize contains the minimum number of bytes required.
0X30000004	The network identifier buffer is not large enough, pNetworkIdentifierSize holds the minimum necessary size in bytes
0X30000005	The operator identifier buffer is not large enough, pOperatorIdentifierSize holds the minimum necessary size in bytes.
0X30000006	The RFInfoList buffer is not large enough
0X30000007	The IPAddress buffer is not sufficient to hold the address. IPAddressSize contains the minimum number of bytes required.
0X30000008	The pFile buffer was insufficient; pFileSize contains the minimum number of bytes required.
0X30000009	The buffer is insufficient. pScanListSize contains the minimum number of bytes necessary to hold the scan list.
0X3000000A	The NAI buffer is insufficient. pNAISize contains the minimum number of bytes required.
0X3000000B	The address buffer is insufficient. The size parameter contains the minimum required byte size.
0X3000000C	The primary ha address buffer is insufficient. The size parameter contains the minimum required byte size.
0X3000000D	The secondary ha address buffer is insufficient. The size parameter contains the minimum required byte size.
0X3000000E	The description buffer needs to be larger; the description length is set to the minimum

	number of bytes required.
0X3000000F	The unique identifier buffer needs to be larger; the unique identifier length is set to the minimum number of bytes required.
0X30000010	The manufacturer name buffer is not large enough.
0X30000011	The Model buffer is not large enough.
0X30000012	The device name buffer is not large enough.
0X30000013	The IMSI buffer is not large enough
0X30000014	The NAA name buffer is not large enough
0X30000015	The MDN buffer is not large enough
0X30000016	The IMEI buffer is not large enough
0X30000017	The ESN buffer is not large enough.
0X30000018	The MEID buffer is not large enough
0X30000019	The MSISDN buffer is not large enough
0X3000001A	The FWVersion buffer is not large enough
0X3000001B	Frequency Band buffer not large enough
0X3000001C	Channel Number UL buffer not large enough
0X3000001D	Channel Number DL buffer not large enough
0X3000001E	The SSID buffer is not large enough. pSSIDSize contains the minimum required buffer size in bytes.
0X3000001F	The BSSID buffer is not large enough. pBSSIDSize contains the minimum required buffer size in bytes.
0X30000020	The pParameters buffer is not large enough. pParametersSize contains the minimum buffer length required.
0X30000021	The pMacAddress buffer is not large enough. pMacAddressSize contains the minimum buffer length required.
0X30000022	The pPINPUKStatusList is not large enough.
0X30000023	The buffer is not large enough to hold the required data. pDataSize is set to the minimum required size in bytes.
0X30000024	The address buffer is not large enough, pAddressSize contains the minimum required size in bytes.
0X30000025	Version buffer is not large enough, pDriverVersionSize contains the required size in bytes.
0X30000026	The pQoSContextList Buffer is not large enough.
0X30000027	The pICCID buffer is not large enough.
0X30000028	The structure is not sufficient to hold the data, the CellularProfileIDListSize will contain the minimum number of bytes required.
0X30000030	The buffer is not sufficient to hold the data, the pHomeNetworkNamelength will contain the minimum number of bytes required.
0X30000031	The buffer is not sufficient to hold the data, the pServingNetworkInfoListSize will contain the minimum number of bytes required.
0X30001000	The size for the pNAAlist buffer is not sufficient, the NAAListsize will contain the number of the elements in the list.

0X30005001	The SMS record buffer is not large enough.
0X30005002	SMSCValue buffer is not large enough
0X30005003	PSIValue buffer is not large enough
0X30005004	The size for the plDList buffer is not sufficient, the plDListSize will contain the number of the elements in the list.
0X30007001	The ServerAddress buffer needs to be larger, The ServerAddressSize is set to the minimum number of bytes required.
0X30007002	The ServerFQDN buffer needs to be larger. The ServerFQDNSize is set to the minimum number of bytes required.
0X30007003	The timestamp buffer is not large enough.
0X30010001	The size of the network structure is not large enough pSize contains the minimum size required.
0X30010002	The address buffer is not large enough, pAddressSize contains the minimum required size in bytes.
0X30010003	Version buffer is not large enough, pSize contains the required size in bytes.
Security Errors	
0XF0000001	The security request supplied when the API was opened does not grant privilege to access this functionality. You may close and reopen the API with updated credentials to perform this operation.
0XF0000002	The authentication failed
0XF0000003	The authentication has been denied. Please seek proper credentials for your access level.
0XF0000004	The security request was malformed. Please consult vendor materials and/or output log.
0XF0000005	The requested access level is not supported

Table 2: Return Values & Error Codes

9.2 UICC Status Words

The following table is listing possible Status Words (SW1 and SW2) provided by the SIM/R-UIM/UICC in accordance with the [ETSI TS 102 221] Status Words list.

Status Words	
Status words (SW1 SW2)	**Description**
90 00	Normal ending of the command
91 XX	Normal ending of the command, with extra information from the proactive UICC containing a command for the terminal. Length 'XX' of the response data
62 00	No information given, state of non volatile memory unchanged
63 CX	Command successful but after using an internal update retry routine 'X' times Verification failed, 'X' retries remaining (For the VERIFY PIN command, SW1SW2 indicates that the command was successful but the PIN was not correct and there are 'X' retries left. For all other commands it indicates the number of internal retries performed by the card to complete the command.)
64 00	No information given, state of non-volatile memory unchanged

65 00	No information given, state of non-volatile memory changed
65 81	Memory problem
67 XX	The interpretation of this status word is command dependent, except for SW2 = '00' (Wrong length)
68 00	No information given
68 81	Logical channel not supported
68 82	Secure messaging not supported
69 00	No information given
69 83	Authentication/PIN method blocked
69 84	Referenced data invalidated
69 89	Command not allowed - secure channel - security not satisfied
6A 81	Function not supported
6A 86	Incorrect parameters P1 to P2
6A 88	Referenced data not found
6B 00	Wrong parameter(s) P1-P2
6E 00	Class not supported
6F XX	The interpretation of this status word is command dependent, except for SW2 = '00' (Technical problem, no precise diagnosis)

Table 3: Status Words Codes

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
OMA-TS-OpenCMAPI-V1_0-20160126-A	26 Jan 2016	Status changed to Approved by TP TP Ref # OMA-TP-2016-0009-INP_OpenCMAPI_V1_0_ERP_for_final_Approval

Appendix B. Static Conformance Requirements (Normative)

The notation used in this appendix is specified in [SCRRULES].

Every API function calls need to be supported by the implementation of the OpenCMAPI. It shall at least support the call of the function and the dedicated generic return value.

But if one the functions is listed as mandatory in one of the following tables the full feature needs to be implemented in the API for the targeted device type.

And if one the functions is listed as Optional in one of the following tables, when implemented then the full feature needs to be implemented in the API for the targeted device type.

B.1 SCR for Mobile Broadband Device

Item	Function	Reference	Requirement
OpenCMAPI-MBD-001-M	Support API Management	7.2	
OpenCMAPI-MBD-002-M	Support Device Discovery APIs	7.3	
OpenCMAPI-MBD-003-M	Support Cellular Network Management APIs	7.4	
OpenCMAPI-MBD-004-M	Support Connection Management APIs	7.5	
OpenCMAPI-MBD-005-M	Support Network Management APIs	7.6	
OpenCMAPI-MBD-006-O	Support CDMA2000 APIs	7.7	
OpenCMAPI-MBD-007-M	Support Device Service APIs	7.8	
OpenCMAPI-MBD-008-M	Support PINs/PUKs Management APIs	7.9	
OpenCMAPI-MBD-009-O	Support UICC Management APIs	7.10	
OpenCMAPI-MBD-010-O	Support WLAN APIs	7.11	
OpenCMAPI-MBD-011-M	Support Statistics APIs	7.12	
OpenCMAPI-MBD-012-M	Support Information Status APIs	7.13	
OpenCMAPI-MBD-013-M	Support SMS Management APIs	7.14	
OpenCMAPI-MBD-014-M	Support USSD Management APIs	7.15	
OpenCMAPI-MBD-015-O	Support GNSS APIs	7.16	
OpenCMAPI-MBD-016-O	Support Data Push Service Management APIs	7.17	
OpenCMAPI-MBD-017-M	Support Callback APIs	8	

B.2 SCR for laptop

Item	Function	Reference	Requirement
OpenCMAPI-LAP-001-M	Support API Management	7.2	
OpenCMAPI-LAP-002-M	Support Device Discovery APIs	7.3	
OpenCMAPI-LAP-003-M	Support Cellular Network Management APIs	7.4	
OpenCMAPI-LAP-004-M	Support Connection Management APIs	7.5	
OpenCMAPI-LAP-005-M	Support Network Management APIs	7.6	
OpenCMAPI-LAP-006-O	Support CDMA2000 APIs	7.7	
OpenCMAPI-LAP-007-M	Support Device Service APIs	7.8	
OpenCMAPI-LAP-008-M	Support PINs/PUKs Management APIs	7.9	
OpenCMAPI-LAP-009-O	Support UICC Management APIs	7.10	
OpenCMAPI-LAP-010-M	Support WLAN APIs	7.11	
OpenCMAPI-LAP-011-M	Support Statistics APIs	7.12	
OpenCMAPI-LAP-012-M	Support Information Status APIs	7.13	
OpenCMAPI-LAP-013-M	Support SMS Management APIs	7.14	
OpenCMAPI-LAP-014-M	Support USSD Management APIs	7.15	
OpenCMAPI-LAP-015-O	Support GNSS APIs	7.16	
OpenCMAPI-LAP-016-O	Support Data Push Service Management APIs	7.17	
OpenCMAPI-LAP-017-M	Support Callback APIs	8	

B.3 SCR for wireless router

Item	Function	Reference	Requirement
OpenCMAPI-WIR-001-M	Support API Management	7.2	
OpenCMAPI-WIR-002-M	Support Device Discovery APIs	7.3	
OpenCMAPI-WIR-003-M	Support Cellular Network Management APIs	7.4	
OpenCMAPI-WIR-004-M	Support Connection Management APIs	7.5	
OpenCMAPI-WIR-005-M	Support Network Management APIs	7.6	
OpenCMAPI-WIR-006-O	Support CDMA2000 APIs	7.7	
OpenCMAPI-WIR-007-M	Support Device Service APIs	7.8	
OpenCMAPI-WIR-008-M	Support PINs/PUKs Management APIs	7.9	
OpenCMAPI-WIR-009-O	Support UICC Management APIs	7.10	
OpenCMAPI-WIR-010-O	Support WLAN APIs	7.11	
OpenCMAPI-WIR-011-M	Support Statistics APIs	7.12	
OpenCMAPI-WIR-012-M	Support Information Status APIs	7.13	
OpenCMAPI-WIR-013-M	Support SMS Management APIs	7.14	
OpenCMAPI-WIR-014-O	Support USSD Management APIs	7.15	
OpenCMAPI-WIR-015-O	Support GNSS APIs	7.16	
OpenCMAPI-WIR-016-O	Support Data Push Service Management APIs	7.17	
OpenCMAPI-WIR-017-M	Support Callback APIs	8	

B.4 SCR for M2M device

Item	Function	Reference	Requirement
OpenCMAPI-M2M-001-M	Support API Management	7.2	
OpenCMAPI-M2M-002-M	Support Device Discovery APIs	7.3	
OpenCMAPI-M2M-003-M	Support Cellular Network Management APIs	7.4	
OpenCMAPI-M2M-004-M	Support Connection Management APIs	7.5	
OpenCMAPI-M2M-005-M	Support Network Management APIs	7.6	
OpenCMAPI-M2M-006-O	Support CDMA2000 APIs	7.7	
OpenCMAPI-M2M-007-M	Support Device Service APIs	7.8	
OpenCMAPI-M2M-008-M	Support PINs/PUKs Management APIs	7.9	
OpenCMAPI-M2M-009-O	Support UICC Management APIs	7.10	
OpenCMAPI-M2M-010-O	Support WLAN APIs	7.11	
OpenCMAPI-M2M-011-M	Support Statistics APIs	7.12	
OpenCMAPI-M2M-012-M	Support Information Status APIs	7.13	
OpenCMAPI-M2M-013-M	Support SMS Management APIs	7.14	
OpenCMAPI-M2M-014-O	Support USSD Management APIs	7.15	
OpenCMAPI-M2M-015-O	Support GNSS APIs	7.16	
OpenCMAPI-M2M-016-O	Support Data Push Service Management APIs	7.17	
OpenCMAPI-M2M-017-M	Support Callback APIs	8	

B.5 SCR for Smart Phone

Item	Function	Reference	Requirement
OpenCMAPI-SMA-001-M	Support API Management	7.2	
OpenCMAPI-SMA-002-M	Support Device Discovery APIs	7.3	
OpenCMAPI-SMA-003-M	Support Cellular Network Management APIs	7.4	
OpenCMAPI-SMA-004-M	Support Connection Management APIs	7.5	
OpenCMAPI-SMA-005-M	Support Network Management APIs	7.6	
OpenCMAPI-SMA-006-O	Support CDMA2000 APIs	7.7	
OpenCMAPI-SMA-007-M	Support Device Service APIs	7.8	
OpenCMAPI-SMA-008-M	Support PINs/PUKs Management APIs	7.9	
OpenCMAPI-SMA-009-M	Support UICC Management APIs	7.10	
OpenCMAPI-SMA-010-M	Support WLAN APIs	7.11	
OpenCMAPI-SMA-011-M	Support Statistics APIs	7.12	
OpenCMAPI-SMA-012-M	Support Information Status APIs	7.13	
OpenCMAPI-SMA-013-M	Support SMS Management APIs	7.14	
OpenCMAPI-SMA-014-M	Support USSD Management APIs	7.15	
OpenCMAPI-SMA-015-O	Support GNSS APIs	7.16	
OpenCMAPI-SMA-016-M	Support Data Push Service Management APIs	7.17	
OpenCMAPI-SMA-017-M	Support Callback APIs	8	

B.6 SCR for Tablets

Item	Function	Reference	Requirement
OpenCMAPI-TAB-001-M	Support API Management	7.2	
OpenCMAPI-TAB-002-M	Support Device Discovery APIs	7.3	
OpenCMAPI-TAB-003-M	Support Cellular Network Management APIs	7.4	
OpenCMAPI-TAB-004-M	Support Connection Management APIs	7.5	
OpenCMAPI-TAB-005-M	Support Network Management APIs	7.6	
OpenCMAPI-TAB-006-O	Support CDMA2000 APIs	7.7	
OpenCMAPI-TAB-007-M	Support Device Service APIs	7.8	
OpenCMAPI-TAB-008-M	Support PINs/PUKs Management APIs	7.9	
OpenCMAPI-TAB-009-M	Support UICC Management APIs	7.10	
OpenCMAPI-TAB-010-M	Support WLAN APIs	7.11	
OpenCMAPI-TAB-011-M	Support Statistics APIs	7.12	
OpenCMAPI-TAB-012-M	Support Information Status APIs	7.13	
OpenCMAPI-TAB-013-M	Support SMS Management APIs	7.14	
OpenCMAPI-TAB-014-M	Support USSD Management APIs	7.15	
OpenCMAPI-TAB-015-O	Support GNSS APIs	7.16	
OpenCMAPI-TAB-016-M	Support Data Push Service Management APIs	7.17	
OpenCMAPI-TAB-017-M	Support Callback APIs	8	

B.7 SCR for Cloud Devices

Item	Function	Reference	Requirement
OpenCMAPI-CLD-001-M	Support API Management	7.2	
OpenCMAPI-CLD-002-M	Support Device Discovery APIs	7.3	
OpenCMAPI-CLD-003-M	Support Cellular Network Management APIs	7.4	
OpenCMAPI-CLD-004-M	Support Connection Management APIs	7.5	
OpenCMAPI-CLD-005-M	Support Network Management APIs	7.6	
OpenCMAPI-CLD-006-O	Support CDMA2000 APIs	7.7	
OpenCMAPI-CLD-007-M	Support Device Service APIs	7.8	
OpenCMAPI-CLD-008-M	Support PINs/PUKs Management APIs	7.9	
OpenCMAPI-CLD-009-M	Support UICC Management APIs	7.10	
OpenCMAPI-CLD-010-M	Support WLAN APIs	7.11	
OpenCMAPI-CLD-011-M	Support Statistics APIs	7.12	
OpenCMAPI-CLD-012-M	Support Information Status APIs	7.13	
OpenCMAPI-CLD-013-M	Support SMS Management APIs	7.14	
OpenCMAPI-CLD-014-M	Support USSD Management APIs	7.15	
OpenCMAPI-CLD-015-O	Support GNSS APIs	7.16	
OpenCMAPI-CLD-016-O	Support Data Push Service Management APIs	7.17	
OpenCMAPI-CLD-017-M	Support Callback APIs	8	

Appendix C. Typical scenario for use of OpenCMAPI in Mobile Broadband - Laptop context (Informative)

C.1 Typical Scenario in laptop environment – Installation user experience

1. The user plug in the USB modem into the laptop
2. The installation process starts
3. When the installation is finished, the CM Application is launched
4. The user starts using the CM Application

C.2 Typical Scenario in laptop environment – CM Application device management

A typical scenario for the use of OpenCMAPI in a laptop environment with the possibility of having multiple devices would be:

1. On start-up, the CM Application calls CMAPI_API_Open ()
2. The CM calls CMAPI_Callback_Register () and register for CMAPI_Callback_DeviceChanged and for CMAPI_Callback_DetectDevicesComplete ().
3. The CM Application initiates enumeration of available devices by calling the function CMAPI_Discovery_DetectDevices ().
4. The OpenCMAPI calls the callback CMAPI_Callback_DetectDevicesComplete () which provides a list of available devices.
5. The CM Application opens one or several devices of the available devices with the function CMAPI_Discovery_OpenDevice (pUniqueDeviceIdentifier)
6. When the device has been successfully opened, the CMAPI_Discovery_OpenDevice returns a device handle. The CM Application stores this handle for future use. Example: a system has two available devices, one modem and one WLAN device. The CM Application decides to open both devices; it saves the handles in two different variables: “modemHandle” and “wlanHandle”.
7. The device handle is used to reference the device in all “device related” API function call; example CMAPI_Information_GetPINStatus (modemHandle..) and CMAPI_WLAN_Connect (wlanHandle…)
8. The CMAPI_Callback_DeviceChanged callback is called when the availability of OpenCMAPI devices changes. Example: the modem which was opened in previous step is unplugged. Shortly after it has been unplugged the OpenCMAPI invokes CMAPI_Callback_DeviceChanged with the handle parameter set to “modemHandle” and the devicestate parameter set to “Unplugged”.
9. The CM Application calls CMAPI_CloseDevice (modemHandle) to close the device, it is no longer available and of no interest (it is not mandatory to close it).
10. The same modem is plugged in again. Shortly after it has been plugged in, the OpenCMAPI calls CMAPI_Callback_DeviceChanged with the parameters set to pUniqueDeviceIdentifier and “plugged”. The CM Application calls CMAPI_Discovery_OpenDevice (pUniqueDeviceIdentifier) etc, see step 5. (In this example the handle parameter CMAPI_Callback_DeviceChanged equals 0 since the device is not already opened)
11. The CM Application calls CMAPI_CloseDevice (modemHandle) to close devices since it is no longer available and of no interest (it is not mandatory to close it though).
12. The CM Application calls CMAPI_CloseDevice (0) to close all devices.
13. The CM Application unregisters for callbacks via CMAPI_Callback_Unregister

14. The CM Application calls CMAPI_CloseAPI ().
15. The CM Application exits.

C.3 Typical Scenario in laptop environment - Deployment and Installation

Concerning the deployment and installation of a CM Application for an USB modem, the following steps will typically be done by the CM Application developer:

1. The CM Application developer customizes/configures the generic OpenCMAPI redistributable installer (generic redistribution) to support the targeted devices and the CM Application equipments. To minimize the overall package size some components can be excluded. Components that may be excluded are: WLAN, GPS and CDMA. The generic redistribution includes support for 'all' devices that conforms to the OpenCMAPI. To minimize the overall package size device support for 'unneeded' devices can be excluded.
2. The result of the previous configuration process is a custom OpenCMAPI redistributable installer (custom redistribution) which supports one or several devices. The custom redistribution includes the necessary device drivers and the selected OpenCMAPI components as well as installation logic.
3. The CM Application developer creates an installer which includes the CM Application and the custom redistribution.
4. The CM Application installer is deployed on device memory, the internet or preinstalled on target machines.
5. The custom redistribution installer is typically launched from within the main CM Application installer.

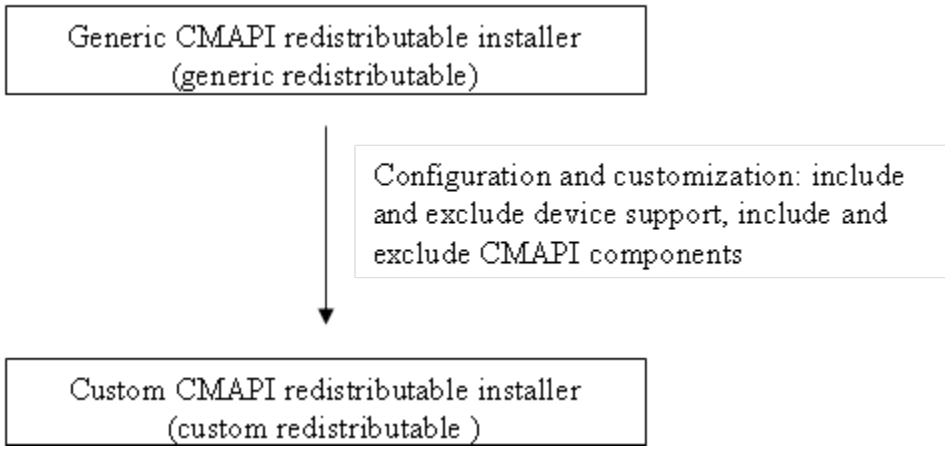

Figure 1: Configuration of OpenCMAPI redistributable installer

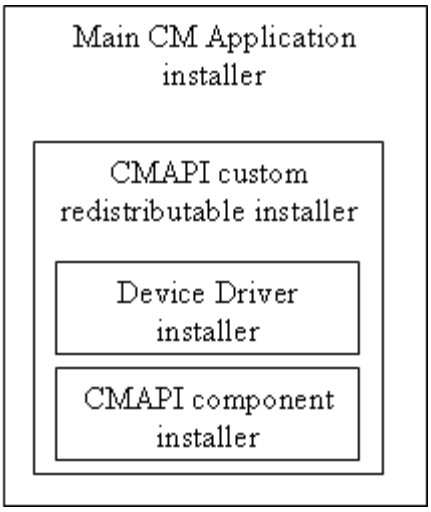

Figure 2: Example of CM Application installer

Appendix D. Consideration for implementation (Informative)

D.1 One Server many clients - Single server

In the “One Server many clients” implementation scenario one single OpenCMAPI server serves many CM clients. An example of this scenario is the built-in Wireless LAN Service in Windows, which serves many applications (Note: it does not mean the OpenCMAPI has to be a part of the OS). In this implementation scenario, the OpenCMAPI is implemented and deployed as a process (an executable application). The communication between client and server relies on a known inter-process communication technique, like Signals, Sockets, Pipes or Message Queues.

D.2 One server per client – Multiple servers

In the “One Server per client” implementation scenario one OpenCMAPI server serves only one CM client. An example of this is vendor specific NDIS API. The NDIS API is implemented in a dll, the CM Application (the client in this aspect) loads the NDISApi.dll into its address space and call functions in the dll. One NDIS API can only serve one client at the time.

D.3 Implementation aspects

D.3.1 Client side aspects

Implementing a CM Application that makes use of a dll (one server per client) is straight forward and is used nearly every application.

Implementing a CM Application that communicates via inter-process (the one server many clients scenario) is not common knowledge and requires a higher level of skill than the dll scenario.

D.3.2 Server side aspects:

One advantage of the single server implementation is that it is possible to share the communication resource (the modem) between several clients. Several CM Applications can for example send SMS in parallel, get the signal strength etc.

It is difficult to implement a shared communication based on the dll scenario.

If the CM Application terminates in an abnormal way, the dll is unloaded automatically by the OS. The underlying communication resources (like COM ports) are also handled automatically by the OS. However in the single server scenario the OS doesn’t handle a crashed client, it has to be done by the SMAPI server itself and is likely to cause problems. In this aspect the dll solution is more reliable.

D.3.3 Deployment

In the single server scenario there will be only one instance installed per system. This can cause problems if one client relies on an ‘old’ server version and another different version. It is easy to maintain and upgrade a system that has a single OpenCMAPI server installed.

In the case of dll, there can be one or several versions of the OpenCMAPI installed on the system. The CM client may install the OpenCMAPI to a common directory or to a private directory. In the case of dll, it is not possible to upgrade all OpenCMAPI servers. Each CM Application has to maintain and upgrade its OpenCMAPI server.

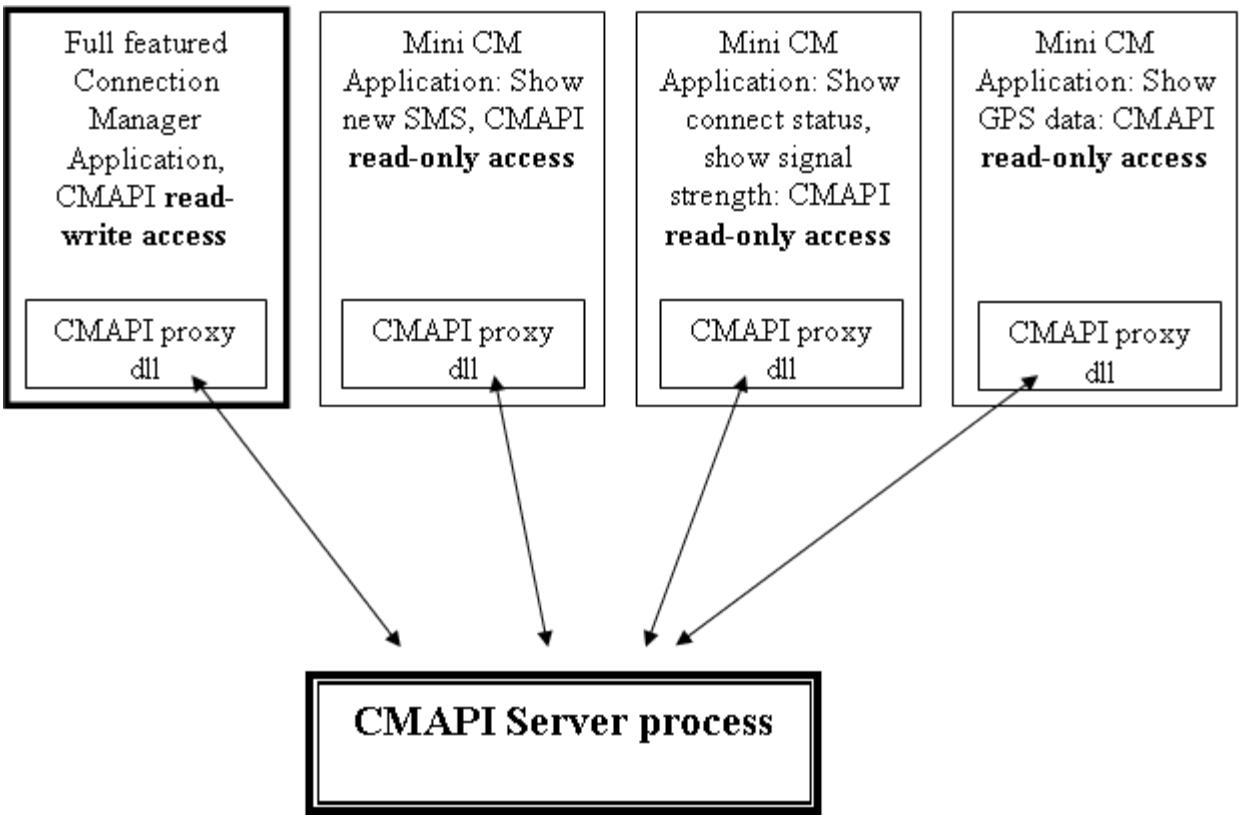

Figure 3: Open CM API as a server process

D.4 Summary

The dll solution is a robust and reliable implementation technique known by 'every' developer. However the dll solution does not offer parallel client sever communication and it is more difficult to maintain and upgrade already deployed applications.

If parallel communication and centralized maintenance and upgrade of deployed CM servers is a strong requirement, then the "**One Server many clients - Single server**" is the best option. In all other cases the dll solution "**One server per client – Multiple servers**" is probably preferable.