Open Connection Manager API

Candidate Version 1.1 – 17 Feb 2015

Open Mobile Alliance

OMA-TS-OpenCMAPI-V1_1-20150217-C

Contents

Figures

Tables

1. Scope

This specification of the OpenCMAPI defines interfaces, 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 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 22.101]	"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 23.040]	"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 23.303]	"TS 23.303 Technical Specification Group Services and System Aspects; Proximity based Services; Stage 2 (Release 12)", 3rd Generation Partnership Project (3GPP), URL: http://www.3gpp.org/ftp/Specs/archive/23_series/23.303/
[3GPP TS 23.402]	"TS 23.402 Technical Specification Group Services and System Aspects; Architecture enhancements for non-3GPP accesses", 3rd Generation Partnership Project (3GPP), URL: http://www.3gpp.org/ftp/Specs/archive/23_series/23.402/
[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 24.229]	"TS 24.229 Technical Specification Group Core Network and Terminals; IP multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP)", 3rd Generation Partnership Project (3GPP), URL: http://www.3gpp.org/ftp/Specs/archive/24_series/24.229/
[3GPP TS 24.234]	"TS 24.234 Technical Specification Group Services and System Aspects; 3GPP system to Wireless Local Area Network (WLAN) interworking; System description", 3rd Generation Partnership Project (3GPP), URL: http://www.3gpp.org/ftp/Specs/archive/24_series/24.234/
[3GPP TS 24.312]	"TS 24.312 Technical Specification Group Core Network and Terminals; Access Network Discovery and Selection Function (ANDSF) Management Object (MO) (Release 12)", 3rd Generation Partnership Project (3GPP), URL: http://www.3gpp.org/ftp/Specs/archive/24_series/24.312/
[3GPP TS 24.334]	"TS 24.334 Technical Specification Group Core Network and Terminals; Proximity-services (ProSe) User Equipment (UE) to Proximity-services (ProSe) Function Protocol aspects; Stage 3; (Release 12)", 3rd Generation Partnership Project (3GPP), URL: http://www.3gpp.org/ftp/Specs/archive/24_series/24.334/
[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 25.331] “TS 25.331 Technical Specification Group Radio Access Network; Working Group 2 (WG2); RRC Protocol Specification”, 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/25_series/25.331/

[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 36.331] “TS 36.331 Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA) Radio Resource Control (RRC); Protocol specification”, 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/36_series/36.331/

[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.S0005] “Upper Layer (Layer 3) Signaling Standard for cdma2000 Spread Spectrum Systems”, 3rd Generation Partnership Project 2 (3GPP2), Technical Specification 3GPP2 C.S0005,
URL: http://www.3gpp2.org/

[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.S0024-0] “cdma2000 High Rate Packet Data Air Interface Specification”, 3rd Generation Partnership Project 2 (3GPP2), Technical Specification 3GPP2 C.S0024-0,
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/

[3GPP2 C.S0069]	"ISIM Application on UICC for cdma2000 Spread Spectrum Systems", 3rd Generation Partnership Project 2 (3GPP2), Technical Specification 3GPP2 C.S0069, URL: http://www.3gpp2.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 102 220]	"TS 102 220 Smart Cards; ETSI numbering system for telecommunication application providers", 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
[IEEE 802.11-2012]	"Telecommunications and information exchange between systems Local and metropolitan area networks--Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", IEEE Standard for Information technology URL: http://standards.ieee.org/about/get/802/802.11.html
[ISO/IEC 14443-3]	"ISO/IEC 14443-3. Identification cards – Contactless Integrated Circuit Cards – Proximity Cards", ISO URL: http://www.iso.org
[ISO/IEC 7816-4]	"Identification cards – Integrated Circuit Cards with contacts - Part 4", ISO URL: http://www.iso.org
[ISO/IEC 7816-5]	"Identification cards – Integrated Circuit Cards with contacts - Part 5", ISO URL: http://www.iso.org
[OpenCMAPI_TS WebAPI]	"Open Connection Manager WebAPI", Open Mobile Alliance™, OMA-TS-OpenCMAPI_Web_V1_1, URL: http://www.openmobilealliance.org/
[OpenCMAPI-AD]	"Open Connection Manager API Architecture", Open Mobile Alliance™, OMA-AD-OpenCMAPI-V1_1, URL:http://www.openmobilealliance.org/
[OpenCMAPI-RD]	"Open CM API Requirements", Open Mobile Alliance™, OMA-RD-OpenCMAPI-V1_1, 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
[RFC4122]	"A Universally Unique IDentifier (UUID) URN Namespace", P. Leach, Microsoft..., July 2005, URL: http://www.ietf.org/rfc/rfc4122.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/
[Wi-Fi Alliance HS2.0 TS]	Hotspot 2.0 (Release 2) Technical Specification version 1.0.0, Wi-Fi Alliance Wi-Fi CERTIFIED Passpoint™ (Release 2) program, URL: https://www.wi-fi.org/ Hotspot_2-0_(R2)_Technical_Specification_v1-0-0.pdf
[Wi-Fi Alliance P2P TS]	Wi-Fi Peer-to-Peer (P2P) Technical Specification v1.5, Wi-Fi Alliance Wi-Fi CERTIFIED Wi-Fi Direct® program URL: https://www.wi-fi.org/Wi-Fi_P2P_Technical_Specification_v1.5.pdf

2.2 Informative References

[OMADICT]	"Dictionary for OMA Specifications", Version 2.9, Open Mobile Alliance™, OMA-ORG-Dictionary-V2_9, 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

For the purpose of this TS, all definitions from the OMA Dictionary apply [OMADICT].

[Secure Element (SE), GP]	“GlobalPlatform Device Technology, Card Specification”, GlobalPlatform™, URL: http://www.globalplatform.org/specificationscard.asp “A Secure Element (SE) is a tamper resistant component which is used in a device to provide the security, confidentiality, and multiple application environments required to support various business models. Such a Secure Element may exist in any form factor such as UICC, embedded SE, smartSD, smart microSD, etc”
AFI	Application Family Indicator as defined in [ETSI TS 102 220] and [ISO/IEC 14443-3]
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.
Connected Device	A client that connects to a router for the purpose of using the router’s data connection to the broadband network
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.
Hotspot 2.0	Hotspot 2.0 [Wi-Fi Alliance HS2.0 TS] (also known as Passpoint) is a set of specifications from the Wi-Fi Alliance.
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.
Local Device	In P2P direct connection context, Local Device will represent the device to be controlled by the OpenCMAPI.
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].

P2P Direct Enabled Device	A device for which P2P (or known as D2D or ProSe in 3GPP) Direct connection is supported and enabled.
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
Remote Device	In P2P direct connection context, Remote Device will represent a device discovered by or communicated to the Local Device.
Router Control Client	Router Control Client represents the application using the WebAPI feature of OpenCMAPI Enabler to manage the Router Device.
Router Device	Router Device represents a device for which the router functionalities are enabled.
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
AFI	Application Family Indicator
AID	Application Identifier
AKA	Authentication and Key Agreement
AN-AAA	Access Network AAA
ANDSF	Access Network Discovery and Selection Function
ANQP	Access Network Query Protocol
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
D2D	Device to Device
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
ESSID	Extended Service Set Identification
ETSI	European Telecommunications Standards Institute
e-UTRAN	evolved Universal Terrestrial Radio Access Network
FDD	Frequency Division Duplex
FQDN	Fully Qualified Domain Name
GAN	Generic Access Network
GERAN	GSM EDGE Radio Access Network
GNSS	Global Navigation Satellite System
GP	Global Platform
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile communications
HA	Home Agent
HCR	High Chip Rate
HESSID	Homogeneous Extended Service Set Identification
HRPD	High Rate Packet Data
HS	HotSpot
HSPA	High Speed Packet Access
IARI	IMS Application Reference ID
IARP	Inter APN Routing Policy
IMPI	IMS Private User Identity
IMPU	IMS Public User Identities
IMS	IP Multimedia Services
IMSI	International Mobile Subscriber Identity
IoT	Internet of Things
ISIM	IMS Identity Module
ISRP	Inter System Routing Policy
LTE	Long Term Evolution
M2M	Machine to Machine
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
MNO	Mobile Network Operator
MO	Management Object
MSID	Mobile Station Identifier
MSISDN	Mobile Station International Subscriber Directory Number
NAA	Network Access Application
NAI	Network Access Identifier
NDIS	Network Driver Interface Specification
NIA	Network-Initiated Alert
NMEA	National Marine Electronics Association
ODM	Original Device Manufacturer
OEM	Original Equipment Manufacturer
OI	Organizational Identifier
OMA	Open Mobile Alliance
OpenCMAPI	Open Connection Manager (CM) Application Programming Interface (API)
P2P	Peer to Peer
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
ProSe	Proximity Services (Also referred to as LTE D2D)
PSK	PreShared Key
PUK	Personal Unlocking Key also called UNBLOCK PIN.
QNC	Quick Net Connect
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
SE	Secure Element
SID	System Identifier
SIM	Subscriber Identity Module
SMS	Short Message Service
SMS-C	Short Message Service Center
SN	Sequence Number
SP	Service Provider
SPC	Service Programming Code
SSID	Service Set Identifier
SSP	Subscription Service Provider
TDD	Time Division Duplex
TLS	Transport Layer Security
TTLS	Tunnelled Transport Layer Security
UE	User Equipment
UI	User Interface
UICC	Universal Integrated Circuit card
UIM	User Identity Module
UMA	Unlicensed Mobile Access
UMTS	Universal Mobile Telecommunications System
URN	Uniform Resource Name
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 interface
  - 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 interface: Callbacks & Registration/Deregistration to receive callbacks

4.2 Version 1.1

Version 1.1 of the Open CM API specification addresses the following aspects and enhancements of the version 1.0 through the different Interfaces:

- CMAPI-1 interface
  - Additional Information Status and statistics functions
  - Phone Book /Contacts management support
  - Extension of WLAN functions including support of Hotspot 2.0, ANDSF & user and operator preferences
  - Support of P2P (or D2D or ProSe as known in 3GPP) Direct connection
  - Router Management support
  - Support of extended device service functions
  - Support of IP Multimedia Services functions
  - Support of dedicated M2M/IoT functions
- CMAPI-2 interface

    - Additional call-backs functions

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(s)
- 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 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF 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**[1]	**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
Extended Device	O	O	O	O	O	O	O

[1] Includes Basic M2M, a subset of M2M representing devices only able to perform basic functions, and, for each group of requirements, only some functions will be supported by Basic M2M.

Service APIs							
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
Contact Management APIs	M	M	M	O	M	M	M
GNSS APIs	O	O	O	O	O	O	O
Data Push Service Management APIs	O	O	O	O	M	M	O
P2P Direct Connection APIs	O	O	O	O	O[2]	O[2]	O
Router Management APIs	O	O	M	O	O	O	O
IMS support APIs	O	O	O	O	O	O	O
M2M/IoT APIs	NA	NA	NA	O	NA	NA	NA
Callback APIs	M	M	M	M	M	M	M
WebAPI	O	O	M	O	O	O	M

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

M – Mandatory

O – Optional

NA – Not Applicable

[2] Strongly Recommended

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,
- dqword will be used to denote 2 qword values,
- byte parameter [256] will indicate a 256 bytes long parameter,
- bit parameter [28] will indicate a 28 bits long parameter,
- 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.
- On a given host, only one instance of the OpenCMAPI Enabler SHOULD be running at a time. All applications, using the OpenCMAPI Enabler SHOULD use and register with this instance.

6.2 Generic Data Type Definitions

6.2.1 CMAPIFunction

Definition CMAPIFunction
This prototype defines an enumeration of group of functions supported by OpenCMAPI.

CMAPIFunction	dword	The following CMAPIFunctions are supported: • 0x00000001: Device Discovery • 0x00000002: Cellular Network Management • 0x00000004: Connection Management • 0x00000008: Network Management • 0x00000010: CDMA2000 • 0x00000020: Device Service • 0x00000040: Device Extended Service • 0x00000080: PIN/PUK Management • 0x00000100: UICC Management • 0x00000200: WLAN • 0x00000400: Statistics • 0x00000800: Information Status • 0x00001000: SMS Management • 0x00002000: USSD Management • 0x00004000: GNSS • 0x00008000: Data Push service Management • 0x00010000: Contact Management • 0x00020000: P2P Direct Management • 0x00040000: Router Management • 0x00080000: IMS services support • 0x00100000: M2M/IoT

6.2.2 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

6.2.3 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)

6.2.4 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.

6.2.5 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

6.2.6 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

6.2.7 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 [RFC5952] and [RFC4291]. Null pointer if no IPaddress is available

6.2.8 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: Signalling 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 signalling 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

6.2.9 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 Bidirectional (Up & Downlink).

6.2.10 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	TrafficFlowTemplate Type*	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].

6.2.11 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) • 0x00000004: IPv6 • 0x00000008: X.25
APN	UTF8*	Primary APN used for this connection
APN2	UTF8*	Optional – Secondary APN
APN3	UTF8*	Optional – Tertiary APN
AccessNumber	UTF8*	(Optional) Access number - sequence to dial to establish the connection (*99# or *99***1# are commonly used as default) Note: this sequence is optional as it may be asked in “old” network using 2G technologies for example
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	SecondaryContextType*	Optional - 7th Secondary Context (if a null pointer value then no 7th SecondaryContext)
SecondaryContext8	SecondaryContextType*	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)
isWlanAllowed	dword	To indicate if the profile is allowed to use WLAN (in case of offload for example – depending on dedicated service) or not: • 0x00000000: Not allowed to use WLAN • 0x00000001: Allowed to use WLAN
maintainCellular	dword	To indicate if the PS connection shall be maintained during WLAN Access • 0x00000000: do not maintain PS connection • 0x00000001: Keep PS connection during WLAN Access

6.2.12 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

6.2.13 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] , [3GPP TS 51.014] 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
pMMSAlertData	MMSAlertData*	[Optional] The raw data of MMS alert. See the definition of MMSAlertData for more details. Note: This field SHALL be available only when msgType is 0x00000002 (MMS alert). The field value SHALL be set to NULL if not available
pVoiceMailData	VoiceMailData*	[Optional] The content of voice mail. See the definition of VoiceMailData for more details. Note: This field SHALL be available only when msgType is 0x00000003 (voice mail). The field value SHALL be set to NULL if not available.
pRAWData	byte*	[Optional] The first byte of pRAWData means the length of the following content in bytes. And the following content is the original RAW data of the Transfer/Transport Layer Message. See Clause 9.2 in TS 23.040 of 3GPP or Clause 3 in C.S0015-A of 3GPP2. Note: This field SHALL be available only when msgType is unknown. The field value SHALL be set to NULL if not available

6.2.14 SMSCreateRecord

Definition SMSCreateRecord
This prototype defines a structure which describes a SMS record for creation purpose.

Field Name	Type	Description
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
pPhoneNumber	UTF8*	The address. See the definition of pPhoneNumber in SMSRecord.
pMsgContent	UTF8*	The text of the message coded in UTF8.
time	UTF8*	The time (local time) when the message was created. The time format should follow : YYYY-MM-DD HH:MM:SS This adheres to ISO 8601

6.2.15 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.

6.2.16 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).

6.2.17 MMSAlertData

Definition MMSAlertData
This prototype defines a structure which describes the raw data of MMS alert.

Field Name	Type	Description
dataSize	dword	The size of the content data in bytes.
data	byte*	The raw data of MMS alert. The raw data here means 1. In the case of 3GPP, the user data of TPDU excluding UDH fields. See 3GPP TS 23.040 2. In the case of 3GPP2, CHAR is of User Data in Bearer Data Sub parameters, The UDHs, if any, SHALL be omitted. See 3GPP2 C.S0015 Note: The API implementer SHALL combine the concatenated segments of MMS, if any, into this field. All the UDHs of SMS shall be omitted.

6.2.18 VoiceMailMessageItem

Definition VoiceMailMessageItem
This prototype defines a structure which describes the voice mail message item.

Field Name	Type	Description
msgID	dword	This parameter shall be set to the message ID of the Voice Mail message in this specific Voice Message notification.
msgLength	dword	This parameter shall be set to the length of the Voice Mail message in this notification in seconds.
retentionDays	dword	This parameter shall be set to the number of days after which the specific Voice Mail message in this notification is anticipated to be automatically deleted from the Voice Mail system timed from the GSM Timestamp for this notification. NOTE: The GSM Timestamp is the time that the SC received the SM from the Voice Mail system which is not necessarily the time that the voice message was deposited into the Voice Mail system.
priority	dword	To indicate the urgency of the voice mail message. • 0x00000000: normal • 0x00000001: interactive, Only available in 3GPP2 system • 0x00000002: urgent • 0x00000003: emergency. Only available in 3GPP2 system
callLineIdentity	UTF8*	To indicate the address to be used to contact the originator of the specific voice mail message in this notification. If the call line identity is not available then the field value SHALL be NULL.
passwordRequired	dword	[Optional] Password required flag. Only available for 3GPP2 system. It indicates whether the password is required or not by the voice mailbox associated with the voice mail message. • 0x00000000: not required • 0x00000001: required • 0xFFFFFFFF: unknown Note: The field value SHALL be set to unknown if not available.
setupRequired	dword	[Optional] Setup required flag. Only available for 3GPP2 system. It indicates whether the mobile station is to prompt the user for initial setup of greeting, password, and recorded name. • 0x00000000: not required

		• 0x00000001: required • 0xFFFFFFFF: unknown Note: The field value SHALL be set to unknown if not available.
passwordChangeRequired	dword	[Optional] Password change required flag. Only available for 3GPP2 system. It indicates whether the user is required or not to change the password before playing the voice mail message. • 0x00000000: not required • 0x00000001: required • 0xFFFFFFFF: unknown Note: The field value SHALL be set to unknown if not available.
minPasswordLen	dword	[Optional] Minimum number of digits required in the password. Only available for 3GPP2 system when either passwordRequired is 0x00000001 or passwordChangeRequired is 0x00000001 Note: The field value SHALL be set to 0 if not available.
maxPasswordLen	dword	[Optional] Maximum number of digits required in the password. Only available for 3GPP2 system when either passwordRequired is 0x00000001 or passwordChangeRequired is 0x00000001 Note: The field value SHALL be set to 0 if not available.
replyAllowed	dword	[Optional] To indicate whether a reply to the voice mail message is allowed or not. Only available for 3GPP2 system. • 0x00000000: not allowed • 0x00000001: allowed • 0xFFFFFFFF: unknown Note: The field value SHALL be set to unknown if not available.
faxIncluded	dword	[Optional] To indicate whether a fax is included or not with the voice mail message. • 0x00000000: not included • 0x00000001: included • 0xFFFFFFFF: unknown Note: The field value SHALL be set to unknown if not available.

6.2.19 VoiceMailMessages

Definition VoiceMailMessages
This prototype defines a structure which describes the voice mail messages.

Field Name	Type	Description
count	dword	The number of the voice mail message items. Note: The field value SHALL be set to 0 if not available.
pMsgItems	VoiceMailMessageItem*	The list of the voice mail message items. Note: The field value SHALL be set to NULL if not available.

6.2.20 VoiceMailNotification

Definition VoiceMailNotification
This prototype defines a structure which describes the content of voice mail notification.

Field Name	Type	Description
mask	dword	A mask to the following available fields. The bitmap definition is • 0x00000001: pMsgContent • 0x00000002: isDiscarded • 0x00000004: isActive • 0x00000008: profileID • 0x00000010: count • 0x00000020: voiceMailboxStatus • 0x00000040: pVoiceMailboxAddress • 0x00000080: voiceMailMessageItems
pMsgContent	UTF8*	[Optional] The content of the message, which was decoded from the user data of TPDU excluding UDHs. Note: The field value SHALL be set to NULL if not available.
isDiscarded	dword	[Optional] To indicate whether the message was discarded or not • 0x00000000: Stored Message • 0x00000001: Discarded Message • 0xFFFFFFFF: Unknown Note: The field value SHALL be set to Unknown if not

		available.
isActive	dword	[Optional] To indicate to set indication as active or inactive • 0x00000000: Set indication inactive • 0x00000001: Set indication active • 0xFFFFFFFF: Unknown Note: The field value SHALL be set to Unknown if not available.
profileID	dword	[Optional] To indicate the profile ID of the Multiple Subscriber Profile • 0x00000000: profile ID 1 • 0x00000001: profile ID 2 • 0x00000002: profile ID 3 • 0x00000003: profile ID 4 • 0xFFFFFFFF: Unknown Note: The field value SHALL be set to Unknown if not available.
count	dword	[Optional] The number of messages waiting Note: The field value SHALL be set to 0 if not available.
voiceMailboxStatus	dword	[Optional] To indicate whether the voice mailbox in the voice mail system is almost full or not. • 0x00000000: not full • 0x00000001: almost full • 0x00000002: full • 0xFFFFFFF: unknown Note: The field value SHALL be set to unknown if not available.
pVoiceMailboxAddress	UTF8*	[Optional] To indicate the voice mailbox address. The 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>”) Note: The field value SHALL be set to NULL if not available.
voiceMailMessageItems	VoiceMailMessages*	The detailed voice mail messages.

6.2.21 VoiceMailDeleteConfirmation

Definition VoiceMailDeleteConfirmation
This prototype defines a structure which describes the content of voice mail delete confirmation.

Field Name	Type	Description
mask	dword	A mask to the following available fields. The bitmap definition is • 0x00000001: pMsgContent • 0x00000002: isDiscarded • 0x00000004: isActive • 0x00000008: profileID • 0x00000010: count • 0x00000020: voiceMailboxStatus • 0x00000040: pVoiceMailboxAddress • 0x00000080: numberOfDeletes • 0x00000100: vmDeletedIDs
pMsgContent	UTF8*	[Optional] The content of the message, which was decoded from the user data of TPDU excluding UDHs. Note: The field value SHALL be set to NULL if not available.
isDiscarded	dword	[Optional] To indicate whether the message was discarded or not • 0x00000000: Stored Message • 0x00000001: Discarded Message • 0xFFFFFFFF: Unknown Note: The field value SHALL be set to Unknown if not available.
isActive	dword	[Optional] To indicate to set indication as active or inactive • 0x00000000: Set indication inactive • 0x00000001: Set indication active • 0xFFFFFFFF: Unknown Note: The field value SHALL be set to Unknown if not available.
profileID	dword	[Optional] To indicate the profile ID of the Multiple Subscriber Profile • 0x00000000: profile ID 1 • 0x00000001: profile ID 2 • 0x00000002: profile ID 3 • 0x00000003: profile ID 4 • 0xFFFFFFFF: Unknown Note: The field value SHALL be set to Unknown if not available.
count	dword	[Optional] The number of messages waiting.

		Note: The field value SHALL be set to 0 if not available.
voiceMailboxStatus	dword	[Optional] To indicate whether the voice mailbox in the voice mail system is almost full or not. • 0x00000000: not full • 0x00000001: almost full • 0x00000002: full • 0xFFFFFFFF: unknown Note: The field value SHALL be set to unknown if not available.
pVoiceMailboxAddress	UTF8*	[Optional] To indicate the voice mailbox address. The 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>") Note: The field value SHALL be set to NULL if not available.
numberOfDeletes	dword	To indicate the number of vmDeletedIDs. Note: The field value SHALL be set to 0 if not available.
pVMDeletedIDs	dword*	The list of the deleted voice mail messages' IDs whose deletion is confirmed. Note: The field value SHALL be set to NULL if not available.

6.2.22 VoiceMailData

Definition VoiceMailData
This prototype defines a structure which describes the voice mail data.

Field Name	Type	Description
type	dword	To indicate the type of the voice mail. • 0x00000000: voice mail notification • 0x00000001: voice mail delete confirmation
pNotification	VoiceMailNotification*	[Optional] A pointer to the content of voice mail notification. It is only available when type is 0x00000000. Note: The field value SHALL be set to NULL if not available.

pDeleteConfirm	VoiceMailDeleteConfirmation*	[Optional] A pointer to the content of voice mail delete confirmation. It is only available when type is 0x00000000. Note: The field value SHALL be set to NULL if not available.

6.2.23 ContactRecord

Definition ContactRecord
This prototype defines a structure which describes a contact record.

Field Name	Type	Description
index	dword	The current index of the contact
contactLocation	dword	To indicate where the contact is stored: • 0x00000000: in the SIM/R-UIM/NAA on UICC; • 0x00000001: in the local device; • 0x00000002: in the terminal device, like PC
pPhoneNumbers	UTF8*	The targeted phone number (each digit in decimal format), , each number shall include its TON (Type Of Number) and NPI (Numbering Plan Identification) parameters (see [3GPP TS 24.008]) coded in binary format (3 digits for TON, 4 digits for NPI) and separated by a space (i.e.: “TON NPI PhoneNumber”), more than one number could be included, each of them is separated by ',', and "\0\0" indicates end of the phone number, dynamic memory allocation. Could be empty
pName	UTF8*	The name of the contact
pEmailAddresses	UTF8*	(optional) email address of the contact - more than one email address could be included, each of them is separated by ',', and "\0\0" indicates end of the email address, dynamic memory allocation

6.2.24 ContactRecords

Definition ContactRecords
This prototype defines a structure which describes the number of Contacts in the list.

Field Name	Type	Description
numberOfRecords	dword	The number of contacts in the array
pContacts	ContactRecord*	The list of the contact records

6.2.25 ConnectionRecord

Definition Connection Record
This prototype defines a structure which describes a connection record.

Field Name	Type	Description
APNName	UTF8*	The name of the APN used
startTime	UTF8*	The start time of the connection – Formatted as defined in ISO 8601: YYYY-MM-DD HH:MM:SS
pEndTime	UTF8*	The end time of the connection – Formatted as defined in ISO 8601: YYYY-MM-DD HH:MM:SS
uploaded	qword	The amount of uploaded data during the connection in Byte
downloaded	qword	The amount of downloaded data during the connection in Byte
PLMNName	UTF8*	The name of the PLMN used 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".

6.2.26 ScreenDesc

Definition ScreenDesc
This prototype defines a structure which provides a description of the screen of the device.

Field Name	Type	Description
screenLocation	dword	Indicate if the screen is a front or a rear screen • 0x00000001: front screen • 0x00000002: rear screen • 0x000000FF: Unknown location
screenTouch	dword	Indicate if the screen is a touch or non screen touch • 0x00000001: touch screen • 0x00000002: no touch screen
screenVerticalDefinition	dword	Indicate the number of pixels of the vertical screen definition
screenHorizontalDefinition	dword	Indicate the number of pixels of the horizontal screen definition

6.2.27 CameraDesc

Definition CameraDesc
This prototype defines a structure which provides a Camera description.

Field Name	Type	Description
cameraLocation	dword	Indicate location of the camera: • 0x00000001: front camera • 0x00000002: rear camera • 0x000000FF: Unknown location
cameraResolution	dword	Indicate the resolution of the camera : unit in number of pixels e.g. : 5 Mega Pixels gives 0x004C4B40

6.2.28 MicroDesc

Definition MicroDesc
This prototype defines an enumeration which provides a microphone description.

microlocation	dword	Indicate location of the microphone : • 0x00000001: internal microphone • 0x00000002: external microphone

6.2.29 LoudSpeakerDesc

Definition LoudspeakerDesc
This prototype defines an enumeration which provides a loudspeaker description.

loudspeakerLocation	dword	Indicate the location of the loudspeaker: • 0x00000001: front • 0x00000002: rear • 0x000000FF: Unknown location

6.2.30 KeypadDesc

Definition KeyPadDesc
This prototype defines an enumeration which provides a keypad description.

kpadType	dword	Indicate the type of the keypad: • 0x00000001: Numerical Key Pad • 0x00000002: Full Alpha-Numerical Key Pad • 0x000000FF: Unknown type

6.2.31 DevAttributes

Definition DevAttributes
This prototype defines the device attributes.

Field Name	**Type**	**Description**
numberofScreen	dword	Number of screens.
pScreenDesc[ ]	ScreenDesc*	pScreenDesc is an array of ScreenDesc of size numberofScreen
numberofCamera	dword	Number of cameras
pCameraDesc[ ]	CameraDesc*	pCameraDesc is an array of CameraDesc of size numberofCamera
numberofMicro	dword	Number of microphones
pMicroDesc[ ]	MicroDesc*	pMicroDesc is an array of MicroDesc of size numberofMicro
numberofSpeaker	dword	Number of Loudspeakers
pLoudspeakerDesc[ ]	LoudspeakerDesc*	pLoudspeakerDesc is an array of LoudspeakerDesc of size of numberofSpeaker
numberofKeyPad	dword	Number of Key Pads.
pKeyPadDesc[ ]	KeyPadDesc*	KeyPadDesc is an array of KeyPadDesc of size numberofKeyPad
Battery	dword	Binary value indicating if the device includes a battery or not: • 0x00000000: No battery • 0x00000001: Battery included

6.2.32 NFCMode

Definition NFCMode
This prototype defines a bitmap of modes for NFC.

NFCMode	dword	The following NFC modes/types are supported: • 0x00000000: no NFC support or not available. • 0x00000001: Reader Mode Type A • 0x00000002: Reader Mode Type B • 0x00000004: Reader Mode Type B' • 0x00000008: Reader Mode Type F • 0x00000010: Card Emulation Mode Type A • 0x00000020: Card Emulation Mode Type B • 0x00000040: Card Emulation Mode Type B' • 0x00000080: Card Emulation Mode Mifare • 0x00000100: Card Emulation Mode Type F • 0x00000200: Peer to Peer Mode

6.2.33 ServiceType

Definition ServiceType
This prototype defines a Service Type in the device

ServiceType	dword	Type of Services (this list is not exhaustive): • 0x00000000: No Service or Unknown • 0x00000001: Transport Services • 0x00000002: Payment Services • 0x00000003: Access Control Services • 0x00000004: Telecom Services • 0x00000005: eHealth Services • 0x00000006: Internet Services (Email, …) • 0x00000007: Games Services • 0x00000008: Secure Storage Services • 0x00000009: PKCS#15 Services • 0x0000000A: DRM Services • 0x0000000B: Loyalty Services

		• 0x0000000C: Identification Services • 0x0000000D: eCars Services

6.2.34 AID

Definition of AID
This prototype defines an AID (Application identifier as defined in [ISO/IEC 7816-4] & [ISO/IEC 7816-5]) associated to a service in the device.

AID	dbyte[16]	An AID is composed of a Registered application provider IDentifier (RID) of 5 bytes and a Proprietary application Identifier eXtension (PIX) of up to 11 bytes (AID is 16 bytes maximum). • Some RID/AID are already defined: 0xA000000009: ETSI (RID only see ETSI 101 220 Annex A for allocated PIX) • 0xA000000087: 3GPP • 0xA000000343: 3GPP2 • 0xA000000412: OMA • 0xA000000424: WiMAX Forum • 0xA0000000031010: Visa • 0xA0000000032010: Visa • 0xA0000000032020: Visa • 0xA0000000038010: Visa • 0xA0000000041010: MasterCard • 0xA0000000049999: MasterCard • 0xA0000000043060: MasterCard • 0xA0000000046000: MasterCard • 0xA0000000050001: MasterCard • 0xA00000002501: American Express • 0xA0000000291010: LINK (UK) ATM network • 0xA0000000421010: CB (France) • 0xA0000000422010: CB (France) • 0xA0000000651010: JCB (Japan) • 0xA0000001211010: Dankort (Denmark) • 0xA0000001410001: CoGeBan (Italy) • 0xA0000001523010: Diners Club/Discover • 0xA0000001544442: Banrisul (Brazil) • 0xA00000022820101010: SPAN2 (Saudi Arabia) • 0xA0000002771010: Interac (Canada) • 0xA0000003241010: Discover • 0xA000000333010101: China UnionPay • 0xA000000333010102: China UnionPay • 0xA000000333010103: China UnionPay • 0xA000000359101002800: Girocard (ZKA Germany) • ……

6.2.35 AFI

Definition of AFI
This prototype defines an AFI (Application Family Identifier) as defined in [ISO/IEC 14443-3].

AFI	byte	Bit 0 to 3 – Sub Group/Family Bit 4 to 7 – Application Group/Family: • 0001: Transport (Local transport, Bus, Airline...) • 0010: Financial (Banking, Retail, tickets...) • 0011: Identification - Access control (passport, driving licence...) • 0100: Telecommunication (Public telephony, GSM...) • 0101: Medical (health insurance card...) • 0110: Multimedia (Internet service, pay service...) • 0111: Gaming (casino card, lotto cards...) • 1000: Data storage (Portable files...) • 1001 to 1111: Reserved for future Applications General AFIs: • 0x00000000: All Application Groups and sub-Groups • 0xXXXX0000: All sub-groups of an Application Group/Family X • 0xXXXXYYYY: Only sub Group Y of Application Group/Family X

6.2.36 ServiceClass

Definition ServiceClass
This prototype defines a Service Class structure A ServiceClass is defined by a Type, an AID and optionally an AFI.

Service_Type	ServiceType	ServiceType is the type of Service
Service_AID	AID	Service_AID is the AID of the service application
Service_AFI	AFI	Optional – Application Family Identifier

6.2.37 SEServices

Definition SEServices
This prototype defines a structure of all the Secure Element Services in the device and associated AID

numberofSE	dword	Number of active Secure Elements in the device
numberofServices	dword	Number of active services discovered in the device
pServiceClassList	ServiceClass*	pServiceClassList is an array of ServiceClass

6.2.38 P2PInfoType

Definition P2PinfoType
This prototype defines an enumeration of the P2P technologies supported.

P2PInfoType	dword	The following P2P Direct Technology (ies) types are supported: • 0x00000000: None • 0x00000001: Wi-Fi Direct (could there be several versions) • 0x00000002: reserved for future use • 0x00000004: reserved for future use • 0x00000008: reserved for future use • 0x00000010: reserved for future use • 0x00010000: LTE Direct • 0x00020000: reserved for future use

6.2.39 ServiceRecord

Definition ServiceRecord
This prototype defines a structure which describes a service record. **Informational Note:** In term of this specification, this is an unformatted field. For interpretation of this field please refer to the related specifications [3GPP 23.003], [3GPP 23.303] and [3GPP 24.334].

Field Name	**Type**	**Description**
serviceID	UTF8*	The identifier of a service. In case of ProSe Discovery it corresponds to the ProSe Application ID Name as specified [3GPP 23.003].
appID	UTF8*	The identifier of an application.

rangeClass	UTF8*	Optional; The range class (rough indication of distance for use in ProSe discovery) requested.
metadata	UTF8*	Optional; Metadata of the service, used for DiscoveryMatch() and DiscoveryResolve() only.

6.2.40 ServiceRecords

Definition ServiceRecords
This prototype defines a structure which describes the number of service identifiers in the list.

Field Name	Type	Description
numberOfRecords	dword	The number of service identifiers in the array
ServiceIDs	ServiceRecord*	The list of the service records

6.2.41 DeviceRecords

Definition DeviceRecords
This prototype defines a structure which describes the number of device identifiers in the list.

Field Name	Type	Description
numberOfRecords	dword	The number of device identifiers in the array
RemoteDeviceIDs	dword*	The list of the remote device IDs

6.2.42 RouterConfigType

Definition RouterConfigType
This prototype defines a structure for router configuration data

Field Name	Type	Description
routerID	dword	The ID of this router configuration
network	WLANNetwork	A data structure defining the network. See WLANNetwork type.
numDevices	dword	A read-only field indicating the number of Connected Devices on the network
maxDevices	dword	The maximum number of Connected Devices

isShared	byte	Indicates whether the network is shared: • 0x00: Not shared • 0x01: Shared
modes	dword	A bitmapped value of supported Wi-Fi physical layer standards: • 0x00000000: None • 0x00000001: 802.11a • 0x00000002: 802.11b • 0x00000004: 802.11g • 0x00000008: 802.11n • 0x00000010: 802.11ac • 0x00000020: 802.11ad
DHCPState	byte	Indicates whether DHCP is enabled: • 0x00: Disabled • 0x01: Enabled
DHCPAddress	UTF8*	The IP Address of the DHCP server in the router
DHCPMask	UTF8*	The subnet mask for IP addresses
DHCPRangeStart	UTF8*	The start of the DHCP IP address range
DHCPRangeEnd	UTF8*	The end of the DHCP IP address range
DHCPDefaultLease	dword	The default time in seconds that a client may keep an IP address
DHCPMaxLease	dword	The maximum time in seconds that a client may keep an IP address

6.2.43 ConnectedDevType

Definition ConnectedDevType
This prototype defines an information structure for Connect Devices which connect to the router.

Field Name	Type	Description
name	UTF8*	The friendly name of the Connected Device
MACAddress	UTF8*	The MAC address of the Connected Device
IPAddress	UTF8*	The IPv4 address of the Connected Device
IPv6Address	UTF8*	The IPv6 address of the Connected Device
connectTime	dword	The number of milliseconds the Connected Device has been connected to the router
txBytes	qword	The number of bytes transmitted during this session

rxBytes	qword	The number of bytes received during this session

6.2.44 PolicyType

Definition PolicyType
This prototype defines an information structure for router policies toward Connected Devices

Field Name	Type	Description
MACAddress	UTF8*	The MAC address of a Connected Device
username	UTF8*	The username associated with this Connected Device
txBpsLimit	dword	Maximum allowed transmission rate for this Connected Device. A value of 0 means no maximum.
rxBpsLimit	dword	Maximum allowed receive rate for this Connected Device. A value of 0 means no maximum.
txBytesPolicy	qword	The number of bytes transmitted since the policy went into effect. Must be set to zero each time a policy is put into effect or reset.
rxBytesPolicy	qword	The number of bytes received since the policy went into effect. Must be set to zero each time a policy is put into effect or reset.
rxtxBytesLimit	qword	Maximum bytes allowed to be transmitted and received by this Connected Device. A value of 0 means no maximum.
txBytesLimit	qword	Maximum bytes allowed to be transmitted by this Connected Device. A value of 0 means no maximum.
rxBytesLimit	qword	Maximum bytes allowed to be received by this Connected Device. A value of 0 means no maximum.
recurring	byte	Indicates whether a policy is one-time or recurring. A recurring policy is used to disable and enable a policy over a specified interval. • 0x00: One-time policy • 0x01: Recurring policy
toStartTime	dword	The start time of the first active session of the policy, in minutes, counting from now. This value decreases by 1 every minute until the policy is either expired (recurring=false), or until the policy is reset (recurring=true) in which case it resets to 0. Note that toStartTime will be negative while the policy is active.
duration	dword	The number of minutes this policy is applied. A value of 0 means infinite.
downTime	dword	If recurring is set to true, this value is the timespan in minutes between two consecutive active policy sessions when the policy is not active
connectTimeLimit	dword	Maximum number of minutes this Connected Device can be connected during an active policy session.
roamingBlock	byte	Indicates whether this Connected Device is blocked from using data while the router is roaming on another WWAN network: • 0x00: Not blocked while roaming

		• 0x01: Blocked while roaming
blocked	byte	Indicates whether this Connected Device is currently blocked due to the policy: • 0x00: Not blocked • 0x01: Blocked

6.2.45 RestrictType

Definition RestrictType
This prototype defines an information structure for Connected Device restrictions

Field Name	Type	Description
MACAddress	UTF8*	The MAC address of a Connected Device
restriction	dword	The type of restriction: Indicates whether the network is shared: • 0x00000001: Blocked (can't use data) • 0x00000002: Blacklisted (can't connect to router)

6.2.46 IMPU

Definition IMPU
This prototype defines a structure representing one IMPU (IMS Public User Identity) in accordance with [3GPP TS 31.103] or [3GPP2 C.S0069].

Field Name	Type	Description
length	byte	The length of the IMPU in byte.
IMPU	UTF8*	One IMS Public User Identity of the ISIM.

6.2.47 IMPUs

Definition IMPUs
This prototype defines a structure representing the IMS Public User Identities (one of more) of the ISIM.

Field Name	Type	Description
numberOfRecords	dword	The number of IMPU in the array
IMPUList	IMPU*	The list of IMS Public User Identities (one of more) of the ISIM.

6.2.48 PDPContext

Definition PDPContext
This prototype defines a structure which describes a PDP context.

Field Name	Type	Description
PDPContextID	dword	The numerical identification of the PDP Context. The parameter is local to the modem interface and is used in other PDP context-related functions.
PDPType	dword	The type of PDP (Packet Data Protocol): • 0x00000001: IP • 0x00000002: PPP - PS data over GPRS or UMTS (PS connection with PDP type PPP) • 0x00000004: IPv6 • 0x00000008: X.25
APN	UTF8*	Primary APN used for this connection
PDPaddress	UTF8*	Optional - The address formatted in compliance with [RFC5952] and [RFC4291]. If the value is null or omitted, then a value may be provided by during the PDP startup procedure or a dynamic address will be requested.
dataCompression	byte	Optional - A numeric parameter that controls PDP data compression • 0: Off (default if value is omitted) • 1: On • Other values are reserved.
headerCompression	byte	Optional - A numeric parameter that controls PDP header compression • 0: Off (default if value is omitted) • 1: On • Other values are reserved.

6.3 WLAN Data Type Definitions

6.3.1 WLANEncryptionType

Definition WLANEncryptionType		
This prototype defines an enumeration of Encryption types for WLAN Note: If the device supports HS2.0, some Encryption types like WEP & TKIP are not allowed		
WLANEncryptionType	dword	The following encryption types are supported: • 0x00000001: none • 0x00000002: WEP • 0x00000004: TKIP • 0x00000008: AES/CCMP

6.3.2 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

6.3.3 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’

6.3.4 RSSI

Definition RSSI
This prototype defines a structure which describes a RSSI signal strength information.

Field Name	Type	Description
signalStrengthRaw	dword	The signal strength value in dBm
signalStrengthPercent	dword	The signal strength as a percentage - SHOULD be adjusted to device capabilities.
signalQualityPercent	dword	The signal quality as a percentage - SHOULD be adjusted to device capabilities.

6.3.5 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
pHS2NetworkInformation	NetworkDiscoveryElements*	Optional – only if the AP is supporting Hotspot 2.0 this field is provided
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 (any network provided by operator policies i.e. ANDSF or HS2.0 has this field set to Automatic)
hidden	dword	Specifies if the SSID is being actively broadcast • 0x00000000: SSID is broadcast • 0x00000001: SSID is hidden
securityType	WLANSecurityType	The type(s) 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”)
known	dword	Identifies if this is a known network • 0x00000000: Unknown (default value) • 0x00000001: Known (Known networks are networks SSID or networks identifiers prelisted by the operator or that have already been used/predefined)
firstTimeConnection	dword	Identifies if the network has already been connected to: • 0x00000000: Unknown or never connected (default value) • 0x00000001: Already Connected to once
pPreference	WLANPreferences*	User and Operator preferences

6.3.6 Located_WLANNetwork

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

Field Name	Type	Description
pNetwork	WLANNetwork*	Please see WLANNetwork
frequency	dword	The frequency of the channel used between the device and the access point (in Mhz)
pSignalStrength	RSSI*	The signal strength of the network
pTimestamp	UTF8*	The Timestamp of the last time the network was located. The time format should follow: YYYY-MM-DD HH:MM:SS+HH:MM (-HH:MM). Adheres to ISO 8601

6.3.7 WLANPreferences

Definition WLANPreferences		
This prototype defines a structure which describes the user and operator preferences attached to this network.		
userPreference	dword	Identifies in which user lists the network belongs to: • 0x00000000: Unknown (default value) • 0x00000001: User Preferred Network – corresponding to the User Controlled WLAN specific Identifier List in 3GPP (see [3GPP TS 24.234] and [3GPP TS 31.102]) • 0x00000002: Network Blacklisted (belongs to blacklist – Network cannot be connected to)
userPreferencePriority	byte	Priority of the network if belongs to User Preferred Network List (if not filled, default priority value of 0 (unknown)). Corresponding as well to the priorities associated to the User Controlled WLAN specific Identifier List in 3GPP (see [3GPP TS 24.234] and [3GPP TS 31.102]) The lower the value, the higher the priority. (0 excluded – meaning unknown)

operatorPreference	dword	Identifies in which Operator list the network belongs to: • 0x00000000: Unknown (default value) • 0x00000001: Operator Preferred Network - corresponding to the Operator Controlled WLAN specific Identifier List in 3GPP (see [3GPP TS 24.234] and [3GPP TS 31.102]) • 0x00000002: Restricted or Excluded Network (cannot connect automatically but can be selected by user)
operatorPreferencePriority	byte	Priority of the network if belongs to Operator Preferred Network List (if not filled, default priority value of 0 (unknown)). Corresponding as well to the priorities associated to the User Controlled WLAN specific Identifier List in 3GPP (see [3GPP TS 24.234] and [3GPP TS 31.102]) The lower the value, the higher the priority. (0 excluded – meaning unknown)

6.3.8 AutoConnectionSettings

Definition AutoConnectionSettings
This prototype defines a structure which describes the different settings for auto connection to WLAN.

Field Name	Type	Description
firstTimeConnection	dword	Specifies if Automatic Connection to WLAN is allowed for first time Connection: • 0x00000000: No Automatic Connection to WLAN if it is first time connection to the WLAN Network • 0x00000001: Automatic Connection to WLAN allowed even if first time
mobilitySettings	dword	Specifies the mobility conditions under which Automatic Connection to WLAN is allowed (bitmap): • 0x00000000: No preference (i.e. do not apply) • 0x00000001: Device is in Idle • 0x00000002: Device is in stationary mode • 0x00000004: Walking • 0x00000008: Running • 0x00000010: In moving vehicle (car, train, airplane...) • 0x00000020: Moving but pattern unknown

batterySettings	dword	(Optional) Specifies the lower value of the battery level in percentage under which Automatic Connection to WLAN is not allowed anymore. 0xFFFFFFFF not provided

6.3.9 WLANOperatorSettings

Definition WLANOperatorSettings
This prototype defines a structure which describes WLAN Operator Settings

Field Name	**Type**	**Description**
offloadSupported	dword	Specifies if WLAN offload is supported: • 0x00000000: WLAN Offload not supported • 0x00000001: WLAN Offload Supported for all services • 0x00000002: WLAN Offload Not supported for all services
operatorSetting	dword	Specifies the operator settings regarding which WLAN policies should apply: • 0x00000000: Not specified – no preferences between ANDSF & HS2.0 • 0x00000001: ANDSF Preferred (default) • 0x00000002: HS2.0 Preferred • 0x00000003: ANDSF Only Any other value is interpreted as default value (ANDSF preferred)

6.3.10 WLANUserSettings

Definition WLANOperatorSettings
This prototype defines a structure which describes WLAN User Settings

Field Name	**Type**	**Description**
automaticConnection	dword	Specifies which type of Connection mode is preferred by the user: • 0x00000000: Manual Only – no Automatic Connection to WLAN • 0x00000001: WLAN Preferred – Automatic Connection to WLAN allowed • 0x00000002: Automatic Connection to WLAN

		under conditions
pAutomaticConnectionOptions	AutoConnectionSettings*	Optional (if the Automatic Connection under condition is selected by the user)
HS20DefaultSubscription	UTF8*	Name of the default HS2.0 WLAN operator subscription
ANDSFDefaultSubscription	UTF8*	Optional – Name of the default ANDSF WLAN operator subscription – in case of dual SIM
userSetting	dword	Specifies the user settings regarding which WLAN policies should apply: • 0x00000000: No preference (i.e. operator settings applies) • 0x00000001: Cellular Preferred (i.e. ANDSF preferred) • 0x00000002: WLAN subscription preferred (i.e. HS2.0 Preferred) Any other value means no preferences (equal to 0x00000000)

6.3.11 EF_UWSIDL

Definition EF_UWSIDL
This prototype defines a structure (in accordance with [3GPP TS 31.102]) which provides the user preferred list of WLAN specific identifier (WSID) from the SIM/R-UIM/NAA on UICC for WLAN selection in priority order.

Field Name	Type	Description
length	byte	Length of the WSID – number of bytes of the following field containing the WSID.
pWSID	UTF8*	WSIDs – User preferred list of WLAN specific identifiers (WSID) as defined in [3GPP TS 24.234] organised in priority order (first record indicates the highest priority and the n^{th} record indicates the lowest). Unused bytes are set to 'FF' and are not used either as a part of the value or for length calculation.

6.3.12 EF_OWSIDL

Definition EF_OWSIDL
This prototype defines a structure (in accordance with [3GPP TS 31.102]) which provides the operator preferred list of WLAN specific identifier (WSID) from the SIM/R-UIM/NAA on UICC for WLAN selection in priority order.

Field Name	Type	Description
length	byte	Length of the WSID – number of bytes of the following field containing the WSID.
pWSID	UTF8*	WSIDs – Operator preferred list of WLAN specific identifiers (WSID) as defined in [3GPP TS 24.234] organised in priority order (first record indicates the highest priority and the nth record indicates the lowest). Unused bytes are set to 'FF' and are not used either as a part of the value or for length calculation.

6.4 ANQP Data Type Definitions

6.4.1 ANQPInfoID

Definition ANQPInfoID
This prototype defines an enumeration of Info ID value of the ANQP elements in accordance with [IEEE 802.11-2012]

ANQPInfoID	word	INFO ID and ANQP Element name: • 1-255: HS2.0 ANQP Info ID accordance with [Wi-Fi Alliance HS2.0 TS] • 1: HS Query List • 2: HS Capability List • 3: Operator Friendly Name • 4: WAN Metrics • 5: Connection Capability • 6 : NAI Home Realm Query • 7: Operator Class Indication • 8: OSU providers List • 9 : Reserved • 10: Icon Request • 11: Icon Binary File • 12-255: Reserved • 256: ANQP Query List • 257: ANQP Capability List • 258: Venue Name • 259 : Emergency Call Number • 260: Network Authentication Type • 261: Roaming Consortium • 262: IP Address Type Availability • 263: NAI Realm • 264: 3GPP Cellular Network • 265: AP Geospatial Location

		• 266: AP Civic Location • 267: AP Location Public Identifier URI • 268: Domain Name • 269: Emergency Alert Identifier URI • 270: TDLS Capability • 271: Emergency NAI • 272– 56796: Reserved • 56797: ANQP Vendor Specific • 56798 – 65535: Reserved

6.4.2 ANQPQueryList

Definition ANQPQueryList
This prototype defines a structure which provides the list of elements queried in accordance with [IEEE 802.11-2012].

Field Name	Type	Description
InfoID	word	256
length	word	The Length value is equal to 2 times the number of ANQP Query ID fields.
ANQPQueryIDs	ANQPInfoID*	Each ANQP Query ID field value is corresponding to a InfoID from the ANQP Info ID table

6.4.3 ANQPCapabilityList

Definition ANQPCapabilityList
This prototype defines a structure which provides the list of elements supported in accordance with [IEEE 802.11-2012].

Field Name	Type	Description
InfoID	word	257
length	word	The Length value is equal to 2 times the number of ANQP Capability fields following the Length field plus the sum of the lengths of the ANQP Vendor Specific lists.
ANQPCapabilities	ANQPInfoID*	ANQP Capabilities supported by the AP Each ANQP capability field value is corresponding to a InfoID from the ANQP Info ID table The Info ID for ANQP Capability List element is always included in the ANQP Capability List element returned

ANQPVendor-Specificelements	ANQPInfoID	Optional When an ANQP Vendor Specific element is present in the ANQP Capability List element, the ANQP Vendor Specific element contains the capabilities of that vendor-specific query protocol. The ANQP Vendor Specific element is defined in Vendor Specific element

6.4.4 VenueNameDuple

Definition VenueNameDuple
This prototype defines a structure which describes the Venue Name Duple in accordance with [IEEE 802.11-2012].

Field Name	Type	Description
length	word	Value is equal to 3 plus the number of octets in the Venue Name field.
languageCode	byte[3]	Language used in the Venue Name field encoded as ISO-14962-1997 . The Language Code field is a two or three character language code selected from ISO-639. A two character language code has 0 (“null” in ISO-14962-1997) appended to make it 3 bytes in length.
pVenueName	UTF8*	Venue’s name. The maximum length of this field is 252 bytes.

6.4.5 VenueNameANQPElement

Definition VenueNameANQPQElement
This prototype defines a structure which describes the Venue Name ANQP element in accordance with [IEEE 802.11-2012].

Field Name	Type	Description
InfoID	word	258
length	word	The Length value is equal to 2 plus the number of bytes in Venue Name Duple fields.
pVenue_Info	VenueInfo*	Venue Info as defined in VenueInfo
pVenueNameDuples	VenueNameDuple*	Optional – Venue Name Duples

6.4.6 NetworkAuthenticationType

Definition NetworkAuthenticationType
This prototype defines a structure which describes the Network Authentication Type in accordance with [IEEE 802.11-2012].

Field Name	Type	Description
networkAuthenticationTypeIndicator	word	Network Authentication Type Indicator: • 0 - Acceptance of terms and conditions • 1 - On-line enrollment supported • 2 - http/https redirection • 3 - DNS redirection • 4 – 255 - Reserved
RURLLength	dword	Re-direct URL Length
RURL	UTF8*	Re-direct URL

6.4.7 NetworkAuthenticationTypeANQPElement

Definition NetworkAuthenticationTypeANQPElement
This prototype defines a structure which describes the list of Network Authentication Types supported when ASRA is set to 1 in accordance with [IEEE 802.11-2012].

Field Name	Type	Description
InfoID	word	260
length	word	The Length value is equal to the number and size of the Network Authentication Type Units.
pNetworkAuthenticationTypes	NetworkAuthenticationType*	Optional - the list of Network Authentication Types supported when ASRA is set to 1

6.4.8 OIDuple

Definition OIDuple
This prototype defines a structure which describes a OI (Organizational Identifier) duple in accordance with [IEEE 802.11-2012].

Field Name	Type	Description
length	byte	The Length value is equal to number of bytes in the OI field
OID	OrganizationalIdentifier	Identifies a roaming consortium (group of SSPs with inter-SSP roaming agreement) or a single SSP.

6.4.9 RoamingConsortium

Definition RoamingConsortium
This prototype defines a structure which provides a list of roaming consortium and/or Service Providers identifiers whose networks are accessible though the AP in accordance with [IEEE 802.11-2012].

Field Name	Type	Description
InfoID	word	261
length	word	The Length value is equal to the number and size of OI Duple fields.
OIDuples	OIDuple*	Optional – List of OIs (Organizational Identifiers) contained within the Roaming Consortium element are also included in this ANQP-element.

6.4.10 Authentication Parameter types List

The following table is listing the possible values for the Authentication Parameter types in accordance with [IEEE 802.11-2012].

Authentication Parameter types

ID	Authentication Information	Description	Length (bytes)
0	Reserved		
1	Expanded EAP Method	Expanded EAP Method Subfield Not supported by OpenCMAPI	7
2	Non-EAP Inner Authentication Type	Enum (0 - Reserved, 1 - PAP, 2 - CHAP, 3 - MSCHAP, 4 - MSCHAPV2)	1
3	Inner Authentication EAP Method Type	Value drawn from IANA EAP Method Type Numbers	1
4	Expanded Inner EAP Method	Expanded EAP Method Subfield Not supported by OpenCMAPI	7
5	Credential Type	Enum (1 - SIM, 2 - USIM, 3 - NFC Secure Element, 4 - Hardware Token, 5 - Softoken, 6 - Certificate, 7 - username/password, 8 - none*, 9 - Reserved, 10 - Vendor Specific) *none means server-side authentication only	1

6	Tunneled EAP Method Credential Type	Enum (1 - SIM, 2 - USIM, 3 - NFC Secure Element, 4 - Hardware Token, 5 - Softoken, 6 - Certificate, 7 - username/password, 8 - Reserved, 9 - Anonymous, 10 - Vendor Specific)	1
7 – 220	Reserved		
221	Vendor Specific	Variable Not supported by OpenCMAPI	Variable
222 – 255	Reserved		

Table 2: Authentication Parameter types List

6.4.11 AuthParam

Definition AuthParam
This prototype defines a structure which describes an Authorisation Parameter in accordance with [IEEE 802.11-2012].

Field Name	Type	Description
ID	byte	Indicates the type of authentication information provided (see Table 3: Authentication Parameter types List).
length	byte	The Length value is equal to the length of the Authentication Parameter Value field.
AuthenticationParameterValue	UTF8*	The Authentication Parameter Value indicated by the ID.

6.4.12 EAPMethod

Definition EAPMethod
This prototype defines a structure which describes an EAP Method in accordance with [IEEE 802.11-2012].

Field Name	Type	Description
length	byte	The Length value is equal to 2 plus the length of the Authentication Parameter fields.
EAPM	byte	The EAP method is set to the EAP Type value as given in IANA EAP Method Type Numbers.
authenticationParameterCount	byte	The Authentication Parameter Count indicates how many additional Authentication Parameter fields are specified for the supported EAP Method. If the Authentication Parameters Count field is 0, there are no Authentication Parameters fields present, meaning no additional Authentication Parameters are specified for the EAP Method.

authenticationParameters	AuthParam*	Optional – List of Authentication Parameters

6.4.13 NAIR

Definition NAIR
This prototype defines a structure which describes a NAI (Network Access Identifier) Realm data in accordance with [IEEE 802.11-2012].

Field Name	**Type**	**Description**
NAIRDFLength	word	NAI Realm Data Field Length The Length value is equal to 3 plus the length of the NAI Realm field plus the sum of the lengths of the EAP Method fields.
NAIREncoding	byte	NAI Realm Encoding Bit 0 - The NAI Realm Encoding Type: • Set to 0 to indicate that the NAI Realm in the NAI Realm field is formatted in accordance with IETF RFC 4282. • Set to 1 to indicate it is a UTF-8 formatted character string that is not formatted in accordance with IETF RFC 4282. Bit 1 to 7: Reserved
NAIRLength	byte	NAI Realm Length equal to the length of the NAI Realm field.
NAIRealm	UTF8*	NAI Realm list – one or more NAI Realms formatted as defined in the NAI Realm Encoding Type bit of the NAI Realm Encoding field. If there is more than one NAI Realm in this field, the NAI Realms are delimited by a semi-colon character (i.e., “;”, which is encoded in UTF-8 format as 0x3B). All the realms included in the NAI Realm field support all the EAP methods identified by the EAP Method fields, if present. The maximum length of this field is 255 octets.
EAPMethodCount	byte	EAP Method Count – indicating the number of EAP methods fields for the NAI realm. If the count is 0, there is no EAP method information provided for the NAI realm.
EAPMethods	EAPMethod*	Optional – List of EAP Methods Each EAP Method field contains a set of Authentication Parameters associated with the EAP-Method.

6.4.14 NAIRealmList

Definition NAIRealmList
This prototype defines a structure which describes a NAI (Network Access Identifier) Realm List in accordance with [IEEE 802.11-2012].

Field Name	Type	Description
InfoID	word	263
length	word	The Length value is equal to 2 plus the total length of the NAI Realm Data fields.
NAIRealmCount	word	Specifies the number of NAI realms included in the NAI Realm list.
NAIRealmData	NAIR*	Optional – List of NAI Realm

6.4.15 3GPPCellularNetworkElement

Definition 3GPPCellularNetworkElement
This prototype defines a structure which describes a 3GPP Cellular Network Element in accordance with [IEEE 802.11-2012].

Field Name	Type	Description
InfoID	word	264
length	word	The Length value is equal to the length of the Payload field.
Payload	UTF8*	3GPP Cellular Network ANQP Element contains cellular information such as network advertisement information e.g., network codes and country codes to assist a 3GPP non-AP STA in selecting an AP to access 3GPP networks.

6.4.16 DomainNameField

Definition DomainNameField
This prototype defines a structure which describes a Domain Name Field in accordance with [IEEE 802.11-2012].

Field Name	Type	Description
length	byte	The Length value is equal to the length in byte of the Domain Name field

pDomainName	UTF8*	Domain name compliant with the “Preferred Name Syntax” as defined in IETF RFC 1035. The maximum length of this field is 255 bytes.

6.4.17 DomainNameList

Definition DomainNameList
This prototype defines a structure which provides a list of one or more domain names of the entity operating the IEEE 802.11 access network in accordance with [IEEE 802.11-2012].

Field Name	Type	Description
InfoID	word	268
length	word	The Length value is equal to the number and size of the Domain Name Fields.
pDomainNameFields	DomainNameField*	Optional – List of Domain Name Fields

6.4.18 IPAddressTypeAvailabilityANQPElement

Definition IPAddressTypeAvailabilityANQPElement
This prototype defines a structure which provides information about the availability of IP address version and type that could be allocated after successful association in accordance with [IEEE 802.11-2012].

Field Name	Type	Description
InfoID	word	262
length	word	The Length value is set to 1
IPaddressTypeInfo	byte	Type of the IP address available Bit 0 to Bit 1 – IPv6 Address Type: • 0: Address type not available • 1: Address type available • 2: Availability of the address type not known • 3: Reserved Bit 2 to Bit 7 – IPv4 Address Type: • 0: Address type not available • 1: Public IPv4 address available • 2: Port-restricted IPv4 address available • 3: Single NATed private IPv4 address available • 4: Double NATed private IPv4 address available

		• 5: Port-restricted IPv4 address and single NATed IPv4 address available • 6: Port-restricted IPv4 address and double NATed IPv4 address available • 7: Availability of the address type is not known • 8–63: Reserved

6.4.19 EmergencyCallNumberUnit

Definition EmergencyCallNumberUnit
This prototype defines a structure defining an Emergency Call Number Unit in accordance with [IEEE 802.11-2012].

Field Name	Type	Description
length	byte	The length of the Emergency Call Number in byte.
ECN	UTF8*	An Emergency Call Number.

6.4.20 EmergencyCallNumberUnits

Definition EmergencyCallNumberUnits
This prototype defines a structure representing a list of emergency phone numbers.

Field Name	Type	Description
numberOfRecords	dword	The number of Emergency Call Numbers in the array
CallNumberList	EmergencyCallNumberUnit*	The list of Emergency Call Numbers

6.4.21 EmergencyCallNumber

Definition EmergencyCallNumber
This prototype defines a structure which provides a list of emergency phone numbers, such as directed by a public safety answering point (PSAP), that is used in the geographic location of the Hotspot in accordance with [IEEE 802.11-2012].

Field Name	Type	Description
InfoID	word	259
length	word	The Length value is set to the total length of the Emergency Call Number Unit fields
numbers	EmergencyCallNumberUnits *	The List of Emergency Call numbers

6.4.22 ANQPAnswer

Definition ANQPAnswer
This prototype defines a structure which provides the list of ANQP elements provided in a response to an ANQP query in accordance with [IEEE 802.11-2012]. Only the ANQP Capability List element is always provided by the Access Point.

Field Name	Type	Description
length	word	Length of the ANQP Answer. Value is 1 plus the sum of the lengths of each optional field present in the structure
ACL	ANQPCapabilityList*	The list of information/capabilities that have been configured on an Access Point.
VNAE	VenueNameANQPElement*	(Optional) The venue name (s) of the hotspot
ECN	EmergencyCallNumber*	(Optional) Emergency Call Number(s)
NATAE	NetworkAuthenticationTypeANQPElement*	(Optional) The Network Authentication Type
RC	RoamingConsortium*	(Optional) The Roaming Consortium information
IPATA	IPAddressTypeAvailabilityANQPElement*	(Optional) The IP address type available
NAIRL	NAIRealmList*	(Optional) The NAI Realm
3GPPCNE	3GPPCellularNetworkElement	(Optional) The 3GPP Cellular Network ANQP-element containing cellular information
DNL	DomainNameList	(Optional) The Domain Name(s)

6.4.23 HSQueryList

Definition HSQueryList
This prototype defines a structure which provides the HS2.0 Query List in accordance with [Wi-Fi Alliance HS2.0 TS].

Field Name	Type	Description
InfoID	word	Set to 56797 – Value for ANQP vendor-specific element
length	word	Set to 6 plus the length of the Payload field.
OI	byte[3]	Set to the value 0x 50 6F 9A, as used by the Wi-Fi Alliance.
type	byte	Set to the value 0x11.
subtype	byte	Set to 1 to identify the HS2.0 ANQP-element: HS Query List
reserved	byte	Field to ensure that the header of the ANQP-element is word aligned
HSElementSubtypes	ANQPInfoID*	List of HS Element Subtypes Each HS2.0 ANQP Query ID field value is corresponding to a InfoID from the ANQP Info ID table (value from 1 to 255 for HS2.0 ANQP Elements). Including a Subtype in the HS Query list indicates that the mobile device performing the GAS Query Request is requesting that the HS2.0 element corresponding to that Subtype be returned in the GAS Query Response. The Subtypes included in the HS Query list shall be ordered by monotonically increasing Subtype value.

6.4.24 HSCapabilityList

Definition HSCapabilityList
This prototype defines a structure which provides the HS2.0 Capability List in accordance with [Wi-Fi Alliance HS2.0 TS]. The HS Capability list element is returned in response to a GAS Query Request. When a mobile device discovers a HS2.0 AP, the mobile device can assume that mandatory HS2.0 ANQP-elements are supported by the HS2.0 AP.

Field Name	Type	Description
InfoID	word	Set to 56797 – Value for ANQP vendor-specific element
length	word	Set to 6 plus the length of the Payload field.
OI	byte[3]	Set to the value 0x 50 6F 9A, as used by the Wi-Fi Alliance.
type	byte	Set to the value 0x11.
subtype	byte	Set to 2 to identify the HS2.0 ANQP-element: HS Capability List
reserved	byte	Field to ensure that the header of the ANQP-element is word aligned
HSCapabilities	ANQPInfoID*	HS2.0 ANQP Capabilities supported by the AP. Each HS2.0 ANQP capability field value is corresponding to a InfoID from the ANQP Info ID table (value from 1 to 255 for the HS2.0 ANQP elements)

		If included in the HS Capability list response, it indicates that a query request for that Subtype will return the requested HS element. The Info ID for HS2.0 ANQP Capability List element is always included in the HS Capability List element returned in a GAS Query Response. The list does not include any duplicate Subtypes. The Subtypes returned in the HS Capability list are ordered by increasing Subtype value.

6.4.25 OperatorFriendlyName

Definition OperatorFriendlyName
This prototype defines a structure which provides the Operator Friendly Name in accordance with [Wi-Fi Alliance HS2.0 TS].

Field Name	Type	Description
InfoID	word	Set to 56797 – Value for ANQP vendor-specific element
length	word	Set to 6 plus the length of the Payload field.
OI	byte[3]	Set to the value 0x 50 6F 9A, as used by the Wi-Fi Alliance.
type	byte	Set to the value 0x11.
subtype	byte	Set to 3 to identify the HS2.0 ANQP-element: Operator Friendly Name
reserved	byte	Field to ensure that the header of the ANQP-element is word aligned
pOperatorName	VenueNameDuple*	Operator Name Duple field is identical to the Venue Name Duple except that the Operator Name replaces the Venue Name in the Venue Name field.

6.4.26 WANMetrics

Definition WANMetrics
This prototype defines a structure which provides the WAN Metrics in accordance with [Wi-Fi Alliance HS2.0 TS].

Field Name	Type	Description
InfoID	word	Set to 56797 – Value for ANQP vendor-specific element
length	word	Set to 6 plus the length of the Payload field.
OI	byte[3]	Set to the value 0x 50 6F 9A, as used by the Wi-Fi Alliance.
type	byte	Set to the value 0x11.

subtype	byte	Set to 4 to identify the HS2.0 ANQP-element: WAN Metrics
reserved	byte	Field to ensure that the header of the ANQP-element is word aligned
WANInfo	byte	WAN Info: Bit 0 to Bit 1 – Link Status (indicates the status of the WAN link): • 0 – Reserved • 1 – Link up • 2 – Link down • 3 – Link in test state Bit 2 – Symmetric Link: • Set to 0 to indicate that the uplink speed is different from the downlink speed. • Set to 1 to indicate that the WAN link has the same link speed in the uplink and downlink direction Bit 3 – At Capacity: • Set to 1 to indicate that the WAN link is at capacity and no additional mobile devices will be permitted to associate to the AP • Set to 0 to indicate otherwise Bit 4 to Bit 7 – Reserved: set to 0 on transmission and ignored on reception
downlinkSpeed	dword	Downlink Speed – Estimated value of the WAN Backhaul link current downlink speed in kilobits per second. For backhaul links that do not vary in speed or those for which no accurate estimation can be made, this attribute contains the nominal speed. The maximum value reported by this field is 4,294,967,296 kbps (approximately 4.2Tbit/s); if the backhaul downlink speed is greater than this value, the maximum value is reported. The downlink speed value is set to 0 when the downlink speed is unknown.
uplinkSpeed	dword	Uplink Speed – Estimated value of the WAN Backhaul link's current uplink speed in kilobits per second. For backhaul links that do not vary in speed or those for which no accurate estimation can be made, this attribute contains the nominal speed. The maximum value reported by this field is 4,294,967,296 kbps (approximately 4.2Tbit/s); if the backhaul uplink speed is greater than this value, the maximum value is reported. The uplink speed value is set to 0 when the uplink speed is unknown.
downlinkLoad	byte	Downlink Load – current percentage loading of the downlink WAN connection, scaled linearly with 255 representing 100%, as measured over an interval the duration of which is reported in

		Load Measurement Duration. In cases where the downlink load is unknown to the AP, the value is set to zero.
uplinkLoad	byte	Uplink Load – current percentage loading of the uplink WAN connection, scaled linearly with 255 representing 100%, as measured over an interval the duration of which is reported in Load Measurement Duration. In cases where the uplink load is unknown to the AP, the value is set to zero.
LMD	word	LMD (Load Measurement Duration) – duration over which the Downlink Load and Uplink Load have been measured, in tenths of a second. Where the actual load measurement duration is greater than the maximum value, the maximum value will be reported. The LMD is set to 0 when neither the uplink nor downlink load can be computed. When the uplink and downlink loads are computed over different intervals, the maximum interval is reported

6.4.27 ProtoPortTuple List

The following table is listing the possible values of ProtoPortTuple in accordance with [Wi-Fi Alliance HS2.0 TS].

ProtoPortTuple		
IP Protocol Value	**Port Number Value**	**Description**
1	0	ICMP, used for diagnostics
6	20	FTP
6	22	SSH
6	80	HTTP
6	443	Used by HTTPS and TLS VPNs
6	1723	Used by Point to Point Tunneling Protocol VPNs
6	5060	VoIP
17	500	Used by IKEv2 (IPSec VPN)
17	5060	VoIP
17	4500	May be used by IKEv2 (IPSec VPN)
50	0	ESP, used by IPSec VPNs

Table 3: ProtoPortTuple List

6.4.28 HS20ProtoPortTuple

Definition HS20ProtoPortTuple
This prototype defines a structure which describes status of IP Protocol and port Number in accordance with [Wi-Fi Alliance HS2.0 TS].

Field Name	Type	Description
IPProtocol	byte	IP protocol field in IPv4 packets or the next header field in IPv6 packets.
portNumber	word	Port Number – destination port number used in conjunction with the protocol defined by the IP Protocol field. When port numbers are not used in conjunction with an IP Protocol, the Port Number field is reserved.
status	byte	Indicates the status of the port: • 0 – Closed • 1 – Open • 2 – Unknown • 3-255 – Reserved

6.4.29 ConnectionCapability

Definition ConnectionCapability
This prototype defines a structure which provides the connection status within the hotspot of the most commonly used communications protocols and ports in accordance with [Wi-Fi Alliance HS2.0 TS].

Field Name	Type	Description
InfoID	word	Set to 56797 – Value for ANQP vendor-specific element
length	word	Set to 6 plus the length of the Payload field.
OI	byte[3]	Set to the value 0x 50 6F 9A, as used by the Wi-Fi Alliance.
type	byte	Set to the value 0x11.
subtype	byte	Set to 5 to identify the HS2.0 ANQP-element: Connection Capability
reserved	byte	Field to ensure that the header of the ANQP-element is word aligned
pProtoPortTuples	HS20ProtoPortTuple*	List of Connection status of communications IP protocol and ports.

6.4.30 NAIHRNameData

Definition NAIHRNameData
This prototype defines a structure which describes the NAI Home Realm Name Data in accordance with [Wi-Fi Alliance HS2.0 TS].

Field Name	Type	Description
NAIRealmEncoding	word	NAI Realm Encoding in accordance with [IEEE 802.11-2012] Bit 0 - The NAI Realm Encoding Type: • Set to 0 to indicate that the NAI Realm in the NAI Realm field is formatted in accordance with IETF RFC 4282. • Set to 1 to indicate it is a UTF-8 formatted character string that is not formatted in accordance with IETF RFC 4282. Bit 1 to 7: Reserved
NAIHomeRealmNameLength	word	Length of the NAI Realm Name field.
NAIHomeRealmName	UTF8*	NAI Home Realm Name: one or more NAI Home Realms formatted as defined in the NAI Realm Encoding Type bit of the NAI Realm Encoding field. If there is more than one NAI Realm in this field, the NAI Realms are delimited by a semi-colon character (i.e., ";", which is encoded in UTF-8 format as 0x3B). The maximum length of this sub-field is 255 octets. Mobile device implementations should be aware that home realms are transmitted in this element without confidentiality protection. Mobile devices are not required to use the NAI Home Realm Query element.

6.4.31 NAIHomeRealmQuery

Definition NAIHomeRealmQuery
This prototype defines a structure which provides the NAI Home Realm Query in accordance with [Wi-Fi Alliance HS2.0 TS].

Field Name	Type	Description
InfoID	word	Set to 56797 – Value for ANQP vendor-specific element
length	word	Set to 6 plus the length of the Payload field.
OI	byte[3]	Set to the value 0x 50 6F 9A, as used by the Wi-Fi Alliance.
type	byte	Set to the value 0x11.
subtype	byte	Set to 6 to identify the HS2.0 ANQP-element: NAI Home Realm Query
reserved	byte	Field to ensure that the header of the ANQP-element is word aligned

NAIHomeRealmCount	byte	NAI Home Realm Count indicates the number of NAI Home Realm Name Data fields included in the NAI Home Realm Query.
NAIHRNameDataList	NAIHRNameData*	List of NAI Home Realm Name Data

6.4.32 HS2ANQPAnswer

Definition HS2ANQPAnswer
This prototype defines a structure which provides the list of HS2.0 ANQP elements provided in a response to an HS2.0 ANQP query in accordance with [IEEE 802.11-2012]. Only the HS Capability List element is always provided by the access point.

Field Name	Type	Description
length	word	Length of the HS2 ANQP Answer. Value is 1 plus the sum of the lengths of each optional field present in the element.
HSCL	HSCapabilityList*	The list of information/capabilities that have been configured on an HS 2.0 Access Point.
OFN	OperatorFriendlyName *	(Optional) The list of Operator Friendly Names of the Hotspot Operator.
WM	WANMetrics*	(Optional) The WAN Metrics element provides information about the WAN link connecting an IEEE 802.11 AN and the Internet.
CC	ConnectionCapability*	(Optional) The Connection Capability element provides information on the connection status in the hotspot of the most commonly used communications protocols and ports.
NAIHRQ	NAIHomeRealmQuery*	(Optional) The NAI Home Realm Query element

6.5 HS2.0 Data Type Definitions

6.5.1 WlanRoamingPartner

Definition WlanRoamingPartner
This prototype defines a structure which describes a Wlan Roaming partner record.

Field Name	Type	Description
FQDN_Match	UTF8*	Indication of the match rules with the Fully Qualified Domain Name (FQDN) of an SP in the Roaming Partner List. is the concatenated string "FQDN" || "," || {"includeSubdomains" | "exactMatch"} where ||= concatenation

		and | = or.
priority	byte	Priority of a Home SP in the roaming partner list. The lower the value, the higher the priority. A roaming partner not in this list has a default priority value of 128 (mid range).
pCountry	UTF8*	The encoding shall be one or more, comma delimited (i.e., ",") ISO/IEC 3166-1 2-character country strings or the country-independent value, "*". If there is a country-specific priority in one or more of the {FQDN_Match, Priority, Country} tuples, it shall override the default, country-independent priority. If the country element is not present in the serving AP's beacon or probe response frames, then only default behaviours (indicated by "*") are permitted.

6.5.2 PreferredRoamingPartners

Definition PreferredRoamingPartners
This prototype defines a structure which describes the number of preferred roaming partners in the list.

Field Name	Type	Description
numberOfPRP	dword	The number of Preferred Roaming Partners in the array
pPreferredRoamingPartner	WlanRoamingPartner*	The list of the Preferred Roaming Partners

6.5.3 BackhaulThreshold

Definition BackhaulThreshold
This prototype defines a structure which describes a backhaul threshold information

Field Name	Type	Description
NetworkType	UTF8*	"Home" or "Roaming"
DLBandwidth	dword	Optional Minimum available downlink bandwidth (in bits per second) calculated based on the downlink speed and backhaul load.
ULBandwidth	dword	Optional Minimum available uplink bandwidth (in bits per second) calculated based on the uplink speed and backhaul load.

6.5.4 ProtoPortTuple

Definition ProtoPortTuple
This prototype defines a structure which describes IP Protocol and port Number of an application.

Field Name	Type	Description
IPProtocol	word	IP protocol field in IPv4 packets or the next header field in IPv6 packets.
portNumber	dword	Optional - specifies one or more port numbers used in conjunction with the IP Protocol, required by one or more operator supported applications on the mobile device. Multiple, comma delimited (i.e., ",") port numbers may be defined if required by the application(s); for example: "21,22".

6.5.5 SPolicy

Definition SPolicy
This prototype defines a structure which describes a subscription policy of a Home SP.

Field Name	Type	Description
pPreferredRoamingPartnerList	PreferredRoamingPartners*	Roaming partner list. Any roaming partner not in this list has a default priority value of 128 (mid range).
pMinBackhaulThreshold	BackhaulThreshold*	Policy for the minimum threshold of available backhaul (WAN) parameters at a hotspot network. Policies can be set for uplink bandwidth, downlink bandwidth or both. If uplink or downlink load measurements are not available at a particular hotspot then the respective uplink or downlink policy is not evaluated at that hotspot. When present, this policy should be evaluated for network selection, unless this policy prevents from selecting any access network (e.g., cellular, Wi-Fi, etc.) or no hotspot has a backhaul parameter greater than or equal to the defined thresholds. When enforced, this policy is only applicable for the initial association to an ESS.
pSPExclusionList	UTF8*	Optional List of SSIDs not preferred by the Home SP.

		The Connection Manager Application shall not autonomously select a hotspot operated by a SP listed in the exclusion list, however the user may manually select such a network.
pRequiredProtoportTuple	ProtoPortTuple*	Optional IP Protocol and port Number required by one or more operator supported application on the mobile device, that operator-supported application(s) on the mobile device require to function properly. Note: This policy is set by an operator to ensure that its own applications are functioning when attached to a roaming HS2.0 network. When present, this policy should be evaluated for network selection, unless this policy prevents the mobile device from selecting any access network (e.g., cellular, Wi-Fi, etc.), or if no roaming networks have support for any of the defined IP Protocol and ports numbers, or if the Connection Capability element is not available. When enforced, this policy is only applicable for the initial association to an ESS.
pMaximumBSSLoadvalue	dword	Optional – maximum acceptable BSS Load policy. The purpose of this is to prevent from joining an AP whose channel is overly congested with traffic and/or interference. This shall only be evaluated when in the presence of a home network. When present, this policy should be evaluated by the Connection Manager Application for network selection, unless this policy prevents the mobile device from selecting any access network (e.g., cellular, Wi-Fi, etc.), or no AP in the ESS have its BSSLoad less than the defined MaximumBSSLoadValue threshold A mobile device is permitted to join any AP in an ESS having a channel utilization value (see clause 8.4.2.30 in Error! Reference source not found.) less than the MaximumBSSLoad value. This policy is advisory: If the mobile device cannot find an AP with channel utilization less than the defined MaximumBSSLoadValue leaf, or if BSSLoad is not available, it may ignore this policy.

6.5.6 FQDN

Definition FQDN
This prototype defines an enumeration of FQDN

FQDN	dword	Fully Qualified Domain Name

6.5.7 HomeOI

Definition HomeOI
This prototype defines a structure which describes the Home OI elements.

Field Name	Type	Description
HSPOI	OrganizationalIdentifier	Organizational Identifier (OI) of the Home Service Provider. If the value of HomeOI matches an OI in the Roaming Consortium List advertised by a hotspot operator, successful authentication with that hotspot is possible.
HomeOIRequired	boolean	Determines whether the HomeOI is required. • If the value of HomeOIRequired is true, then no authentication shall be attempted unless the HomeOI value is included in the hotspot's Roaming Consortium List. • If the value of HomeOIRequired is false then authentication should be attempted when the mobile's realm/PLMN ID is advertised by the AP.

6.5.8 HomePartnersList

Definition HomePartnersList
This prototype defines a structure which describes the number of Home Partners in the list.

Field Name	Type	Description
numberOfFQDN	dword	The number of FQDN partner operators in the array
pHomePartners	FQDN*	The list of the FQDN of a partner operator of the Home SP that shall be deemed by the mobile device as a home operator.

6.5.9 HSP

Definition HSP
This prototype defines a structure which describes Home Service Provider elements.

Field Name	Type	Description
pSSID	UTF8*	The service set identifier
pBSSID	UTF8*	The basic service set identifier
pFriendlyName	UTF8*	The FriendlyName is intended to be used by the mobile device for display purposes only.
pIconURL	UTF8*	Optional – location of icon related to the Home SP. The URL is the network location from which the icon can be retrieved and can be used by the mobile device in an implementation dependent manner or used in a manner agreed between the SP and the mobile device manufacturer. The icon is intended to be used by the mobile device for display purposes only.
HFQDN	FQDN	FQDN of the Home SP (to identify if the hotspot is operated by the mobile device's home SP or a visited SP)
pHomeOIList	HomeOI*	Optional – List of organizational identifiers identifying the home SP of which this provider is a member .
pOtherHomePartners	HomePartnersList*	Optional – list of FQDNs of Home partners
pRoamingConsortiumOI	UTF8*	Optional – Contains one or more, comma delimited (i.e., ",") organizational identifiers identifying a roaming consortium of which this provider is a member. Note: The Chr encoding for each OI is in lowercase ASCII hexadecimal characters only with no white space and no preceding "0x", e.g., "506f9a".

6.5.10 SParameters

Definition SParameters
This prototype defines a structure which describes parameters of the subscription.

Field Name	Type	Description
pCreationDate	UTF8*	Date and time (UTC) that the PerProviderSubscription MO was initially provisioned to the mobile device. The date and time is formatted as YYYY-MM-DDTHH:MM:SSZ
pExpirationDate	UTF8*	Optional – Date and time (UTC) that the subscription will expire. After the expiration date, the mobile device should

		not expect to be able to successfully authenticate with the corresponding credentials. This is an optional attribute; if not present, there is no pre-determined expiration time and date.
pTypesofSubscription	UTF8*	Optional – Type of subscription associated with the account. Subscription types are defined by the Home SP and are out of scope of this specification; example values are "Gold", "Silver" and "Bronze".
dataLimit	dword	Optional – Cumulative data limit in megabytes for the UsageTimePeriod. If the value of this parameter is zero, there is unlimited data usage for this account. When the measured amount of data, has been transmitted between the mobile device and the network, reaches this limit,
pStartDate	UTF8*	Optional – Date and time (UTC) at which usage statistics accumulation begins. This parameter specifies the date and time (UTC) that the subscription will become valid. Before the StartDate, the mobile device should not expect to be able to successfully authenticate with the corresponding credentials. This is an optional attribute; if not present, there is no pre-determined start time and date.
timeLimit	dword	Cumulative time limit in minutes for the UsageTimePeriod. If the value of this parameter is zero, there is unlimited time usage for this subscription. When the measured amount of time, used by the mobile device, has reached this limit, the consequences are per the user's subscription
usageTimePeriod	dword	Time period for usage statistics accumulation. A value of zero means that usage statistics are not accumulated on a periodic basis (e.g., a one-time limit for “pay as you go” - PAYG service). A non-zero value specifies the usage interval in minutes. After the expiry of this time, the usage statistics are reset to zero. (e.g., the expiry of a billing period interval would reset the DataLimit). The values of 1 to 31 should be reserved to indicate that usage statistics (e.g., DataLimit) are monthly and reset on the day of the month indicated by the value (e.g., if the value is 10, the usages statistics are kept on a monthly basis and reset on the 10th of each month).

6.5.11 EAPMethodType

Definition EAPMethodType
This prototype defines a structure which describes an EAP Method Type

Field Name	Type	Description
EAPType	EAPAuthenticationMethod	EAP Type value. The possible values are listed in the IANA EAP Registry List
pVendorID	UTF8*	Optional – Vendor-Id for an expanded EAP method, if used.
pVendorType	UTF8*	Optional –Vendor-Type of the expanded EAP method, if used. These values are defined by the vendor identified by VendorId.
innerEAPType	EAPAuthenticationMethod	Optional – EAP Type value for the inner EAP method, if used with this EAP method. The possible values are listed in the IANA EAP Registry.
pInnerVendorID	UTF8*	Optional – Vendor-Id for an inner expanded EAP method, if used.
pInnerVendorType	UTF8*	Optional – Vendor-Type of the inner expanded EAP method, if used
pInnerMethod	UTF8*	Optional – Inner non-EAP method, if used with this EAP method. The permitted values are: PAP, CHAP, MS-CHAP and MS-CHAP-V2.

6.5.12 Certificate

Definition Certificate
This prototype defines a structure which provides information related to the certificate credential.

Field Name	Type	Description
pCertificateType	UTF8*	Certificate type. The value is selected from the following enumerations: “802.1ar” or “x509v3”.
pCertSHA256Fingerprint	UTF8*	SHA-256 fingerprint of the certificate credential for a subscription. This parameter specifies the Issuer Distinguished Name in the certificate credential. In conjunction with the certificate serial number, it uniquely identifies a certificate. This parameter is formatted as follows (defined in XML regular expression syntax): [a-f0-9]{64} i.e. 64 lowercase hexadecimal characters. The SHA-256 fingerprint is calculated over the X.509 ASN.1 DER encoded certificate.

6.5.13 SIMCredential

Definition SIMCredential
This prototype defines a structure which provides information related to the SIM credential.

Field Name	Type	Description
pIMSI	UTF8*	IMSI (International Mobile device Subscriber Identity) (Note: If there are SIM/USIM, OpenCMAPI determines if it should be able to successfully authenticate to a hotspot by comparing the MCC/MNC from its IMSI with the PLMN ID containing a MCC and MNC returned in the 3GPP Cellular Network ANQP-element) Note: the IMSI is included so that the PerProviderSubscription MO can be bound to the correct SIM card in cases where there is more than one SIM card in a mobile device.
pEAPType	UTF8*	EAP Type value. The possible values are listed in the IANA EAP Registry in the Method Types. Only EAP-SIM, EAP-AKA, and EAP-AKA' methods are permitted.

6.5.14 UnPw

Definition UnPw
This prototype defines a structure which provides the username and password values of the credential.

Field Name	Type	Description
pUsername	UTF8*	Username
pPassword	UTF8*	Password (Null Value if there is an application needed to generate the password)
machineManaged	boolean	This parameter specifies whether the password is machine managed. • 0: Not Machine Managed • 1: Machine Managed (It is set to true, if the SP has provided the username and password)
pSoftTokenApp	UTF8*	Optional – Specifies the application that should be used to generate the password. If present, the Password should have a null value
abletoshare	boolean	Optional – indicates whether the credential is usable only on the mobile device which subscribed or usable by other mobile devices of the user as well • 0: Not Able to Share • 1: Able to Share credential
pEAPMethodUsed	EAPMethodType*	EAP Method used

6.5.15 SCredential

Definition SCredential
This prototype defines a structure providing the credentials of the subscription. (Note: Exactly one of the "UsernamePassword", "DigitalCertificate" or "SIM" is present – This is always assured by the Subscription Server).

Field Name	Type	Description
pCreationDate	UTF8*	Date and time (UTC) when the credential was either created or last updated The date and time is formatted as YYYY-MM-DDTHH:MM:SSZ This adheres to ISO 8601
pExpirationDate	UTF8*	Optional - Date and time (UTC) that the credentials will expire. (if not present, there is no pre-determined expiration time and date) The date and time is formatted as YYYY-MM-DDTHH:MM:SSZ This adheres to ISO 8601
pUsernamePassword	UnPw*	Optional - Username and password values of the credential.
pDigitalCertificate	Certificate*	Optional - Information related to the certificate credential.
pRealm	UTF8*	The Realm associated with the credential. (Note: OpenCMAPI determines if it should be able to successfully authenticate to a hotspot by comparing the realms returned in the NAI Realm ANQP-element with this realm)
checkAAAServerCertStatus	boolean	Optional - Indicates whether the device must check the AAA server certificate's revocation status during EAP authentication (i.e., for EAP methods which employ a AAA server certificate). • If the field CheckAAAServerCertStatus is present and set to true, then then mobile devices shall use CSP as defined in section 7.7.3.2 to check the AAA server certificate's revocation status during EAP authentication; • If the field CheckAAAServerCertStatus is not present or is present and set to false then mobile devices shall not require the AAA server certificate's revocation status to be available at the time of authentication.
pSIM	SIMCredential*	Optional - information related to the SIM credential.

6.5.16 HS2Subscription

Definition HS2Subscription
This prototype defines a structure which describes a subscription profile (Home SP information, subscription policy, management and credential information) for Hotspot 2.0.

Field Name	Type	Description
pPolicy	SPolicy*	Optional Home SP policy.
subscriptionPriority	word	Priority of the subscription, when multiple subscriptions are associated with a service provider. The lower the value of priority, the higher the subscription priority.
pHomeSP	HSP*	Home SP information for this subscription.
pSubscriptionParameters	SParameters*	Optional identify the subscription parameters.
pCredential	SCredential*	Credential of the subscription

6.5.17 PerProviderSubscription

Definition PerProviderSubscription
This prototype defines a structure providing one or more HS2.0 subscriptions (these elements are part of the HS2.0 Management Object but only the parts relevant for the OpenCMAPI).

Field Name	Type	Description
updateIdentifier	word	Identifies if there is existing provisioned MO subscription or not • 0x0000: Unprovisioned values in the MO (default) • Other: Existing provisioned MO
numberOfsubscription	dword	The number of subscriptions in the array
pSubscription	HS2Subscription*	The list of subscriptions with their details

6.5.18 VenueGroup

The following table is listing Venue Group Code and description in accordance with the [IEEE 802.11-2012].

Venue Group	
Venue Group Code	**Description**
0	Unspecified

1	Assembly
2	Business
3	Educational
4	Factory and Industrial
5	Institutional
6	Mercantile
7	Residential
8	Storage
9	Utility and Miscellaneous
10	Vehicular
11	Outdoor
12	Personal Network
13 – 255	Reserved

Table 4: Venue Group

6.5.19 VenueType

The following table is listing Venue Type Code and Description in accordance with the [IEEE 802.11-2012].

Venue Type		
Venue Group	**Venue Type Code**	**Description**
0	0	Unspecified
0	1 - 255	Reserved
1	0	Unspecified Assembly
1	1	Arena
1	2	Stadium
1	3	Passenger Terminal (e.g., airport, bus, ferry, train station)
1	4	Amphitheater
1	5	Amusement Park
1	6	Place of Worship
1	7	Convention Center
1	8	Library
1	9	Museum
1	10	Restaurant
1	11	Theater
1	12	Bar
1	13	Coffee Shop
1	14	Zoo or Aquarium

1	15	Emergency Coordination Center
1	16 - 255	Reserved
2	0	Unspecified Business
2	1	Doctor or Dentist office
2	2	Bank
2	3	Fire Station
2	4	Police Station
2	6	Post Office
2	7	Professional Office
2	8	Research and Development Facility
2	9	Attorney Office
2	10 – 255	Reserved
3	0	Unspecified Educational
3	1	School, Primary
3	2	School, Secondary
3	3	University or College
3	4-255	Reserved
4	0	Unspecified Factory and Industrial
4	1	Factory
4	2 – 255	Reserved
5	0	Unspecified Institutional
5	1	Hospital
5	2	Long-Term Care Facility (e.g., Nursing home, Hospice, etc.)
5	3	Alcohol and Drug Re-habilitation Center
5	4	Group Home
5	5	Prison or Jail
5	6 – 255	Reserved
6	0	Unspecified Mercantile
6	1	Retail Store
6	2	Grocery Market
6	3	Automotive Service Station
6	4	Shopping Mall
6	5	Gas Station
6	6 – 255	Reserved
7	0	Unspecified Residential
7	1	Hotel or Motel
7	2	Dormitory

7	3	Boarding House
7	4 – 255	Reserved
8	0 – 255	Reserved
9	0 – 255	Reserved
10	0	Unspecified Vehicular
10	1	Automobile or Truck
10	2	Airplane
10	3	Bus
10	4	Ferry
10	5	Ship or Boat
10	6	Train
10	7	Motor Bike
10	8 – 255	Reserved
11	0	Unspecified Outdoor
11	1	Muni-mesh Network
11	2	City Park
11	3	Rest Area
11	4	Traffic Control
11	5– 255	Reserved
12	0	Reserved

Table 5: Venue Type

6.5.20 VenueInfo

Definition VenueInfo
This prototype defines a structure which provides the information about a venue.

Field Name	**Type**	**Description**
venueGroup	byte	Venue Group in accordance with the corresponding table
venueType	byte	Venue Type in accordance with the corresponding table

6.5.21 OrganizationalIdentifier

Definition OrganizationalIdentifier
This prototype defines an enumeration of OI (Organizational Identifier).

OrganizationalIdentifier	UTF8*	Organizational Identifier (OI) of the Service Provider and Roaming consortiums. Globally unique identifier assigned by the IEEE – similar to the first half of a MAC address. OI is a lowercase ASCII hexadecimal characters only with no white space and no preceding “0x”, e.g., “506f9a” (often 24 bits in length, but can also be 36 bits)

6.5.22 IE

Definition IE
This prototype defines a structure which provides the IE (Interworking Element) in accordance with [IEEE 802.11-2012].

Field Name	Type	Description
elementID	byte	Unique value for each information element. For interworking element, the value is 107
length	byte	Length of the Interworking Element. Value is 1 plus the sum of the lengths of each optional field present in the element.
accessNetworkOptions	byte	Bit 0 to 3: Access Network Type field set by the AP to advertise its Access Network Type: • 0 - Private network • 1 - Private network with guest access • 2 - Chargeable public net-work • 3 - Free public network • 4 - Personal Device Net-work • 5 - Emergency Services Only Network • 6 to 13 - Reserved • 14 - Test or experimental • 15 - Wildcard Bit 4 - the Internet field: • Set to 1 to indicate the AP provides connectivity to the Internet; • Set to 0 to indicate that it is unspecified whether the network provides connectivity to the Internet. Bit 5 – the Additional Step Required for Access (ASRA) field: • Set to 1 by the AP to indicate that the network requires a further step for access.

		Bit 6 – the ESR (Emergency Services Reachable) field: • Set to 1 by the AP to indicate that emergency services are reachable through the AP; • Set to 0 to indicate that it is unspecified whether emergency services are reachable. Bit 7 – the UESA (Unauthenticated Emergency Service Accessible) field. • Set to 1 to indicate that higher layer unauthenticated emergency services are reachable through this AP; • Set to 0 indicating that no unauthenticated emergency services are reachable through this AP.
pVenueInformation	VenueInfo*	Optional Venue Info field is a 2 bytes field that provides information describing the venue containing Venue Group and Venue Type subfields (see VenueInfo structure)
HESSID	byte[6]	Optional The HESSID (Homogeneous ESS ID) is a 6 bytes MAC address that identifies the homogeneous ESS. The HESSID value shall be identical to one of the BSSIDs in the homogeneous ESS. Unique identifier that in conjunction with the SSID, may be used to provide network identification for an SSPN.

6.5.23 APT

Definition APT
This prototype defines a structure which provides the Advertisement Protocol Tuple in accordance with [IEEE 802.11-2012].

Field Name	Type	Description
queryResponseInfo	byte	The Query Response Info in accordance with [IEEE 802.11-2012]. Bit 0 to 6: QueryResponseLength: Maximum number of bytes will be transmitted in the Query Response A value of zero is not permitted. A value of 0x7F means the maximum limit enforced is determined by the maximum allowable number of fragments in the GAS Query Response Bit 7, the Pre-Association Message Exchange BSSID Independent (PAME-BI) Used by an AP to indicate whether the Advertisement server will return a Query Response which is independent of the BSSID used for the GAS frame exchange.

		• Set to 0 indicating that the Query Response may be dependent on the BSSID • Set to 1 indicating that the Query Response is independent of the BSSID
advertisementProtocolID	byte	Advertisement Protocol ID: 0 - Access Network Query Protocol 1 - MIH Information Service 2 - MIH Command and Event Services Capability Discovery 3 - Emergency Alert System (EAS) 4 - Location-to-Service Translation Protocol 5-220 - Reserved 221 - Vendor Specific 222-255 – Reserved For the purpose of OpenCMAPI version 1.1, only Advertisement Protocol ID value set to 0 (Access Network Query Protocol) is relevant

6.5.24 APE

Definition APE
This prototype defines a structure which provides the APE (Advertisement Protocol Element) in accordance with [IEEE 802.11-2012].

Field Name	Type	Description
elementID	byte	For Advertisement Protocol information element, the value is 108
length	byte	Length of the Advertisement Protocol Element. Value is 1 plus the sum of the lengths of each optional field present in the element.
pAPTList	APT*	List of Advertisement Protocol Tuples

6.5.25 RCE

Definition RCE
This prototype defines a structure which provides the RCE (Roaming Consortium Element) in accordance with [IEEE 802.11-2012].

Field Name	Type	Description
elementID	byte	For roaming consortium information element, the value is

		111
length	byte	Length of the Roaming Consortium Element. Value is 1 plus the sum of the lengths of each optional field present in the element.
numberANQPOIs	byte	Number of ANQP OIs field: Number of additional roaming consortium organization identifiers (OIs) obtainable via ANQP. • Value of 0 indicating that no additional OIs will be returned in response to a ANQP query for the Roaming Consortium list. • Value of 255 indicating that 255 or more additional OIs are obtainable via ANQP.
OI1and2Lengths	byte	The OI 1 and 2 Lengths field • Bit 0 to Bit 3 – value of the OI1 Length subfield is the length in bytes of the OI1 field. • Bit 4 to Bit 7 – value of the OI2 Length subfield is the length in bytes of the OI2 field. If the OI2 field is not present, the value of the OI2 Length subfield is set to zero.
OI1	OrganizationalIdentifier	Each OI identifies a roaming consortium (group of Subscription Service Providers (SSP) with inter-SSP roaming agreement) or a single Service Provider. The value of the OI(s) in Roaming Consortium Element are equal to the value of the first 3 OIs in the Roaming Consortium List in response to a ANQP Query. The remainder are available through a ANQP Query. If fewer than 3 values are defined in the Roaming Consortium List, then only as many OIs as defined in the table are populated in this element.
OI2	OrganizationalIdentifier	Optional
OI3	OrganizationalIdentifier	Optional

6.5.26 NetworkDiscoveryElements

Definition NetworkDiscoveryElements
This prototype defines a structure which provides the information elements related to the network discovery in accordance with [IEEE 802.11-2012] & [Wi-Fi Alliance HS2.0 TS].

Field Name	**Type**	**Description**
PasspointIndicationElement	HS20IE*	Indicate support and compliance with HS2.0 certification
pInterworkingElement	IE*	Identifies the interworking service capabilities of the AP
pAdvertisementProtocolElement	APE*	Identifies the AP’s support of particular advertisement protocols (e.g. ANQP)

pRoamingConsortiumElement	RCE*	Identifies the service providers or roaming partners supported by the AP

6.5.27 HS20IE

Definition HS20IE
This prototype defines a structure which provides the HS2.0 Indication Element in accordance with [Wi-Fi Alliance HS2.0 TS] & [IEEE 802.11-2012].

Field Name	Type	Description
ElementID	byte	For HS 2.0 Indication Element, the value is 221
length	byte	Length of the HS 2.0 Indication Element. Value is set to 5 or 7. Note: in Release 2 of [Wi-Fi Alliance HS2.0 TS], the PPS MO ID field or the ANQP Domain ID field (mutually exclusive) is included in the HS2.0 Indication Element.
OI	byte[3]	Set to the value 0x50 6F 9A, as used by the Wi-Fi Alliance.
type	byte	Set to the value 0x10.
HotspotConfiguration	byte	Hotspot Configuration field: Bit 0: DGAF (Downstream Group-Addressed Forwarding) Disabled: - For an Access Point: • Set to 1 when the AP is not forwarding downstream group-addressed frames • Set to 0 Otherwise - For a mobile device, • Set to 0 when the HS2.0 Indication element is included in a (re)association request frame Bit 1: The PPS MO ID Present • Set to 1 when the PPS MO ID field is present in the HS2.0 indication element • Set to 0 Otherwise Bit 2: The ANQP Domain ID Present • Set to 1 when the ANQP Domain ID field is present in the HS2.0 indication element • Set to 0 Otherwise Bit 3: Reserved Bit 4 to Bit 7: Release Number identifying the HS2.0 release capability: • 0 – Release 1

		• 1 – Release 2 • 2 to 15 - Reserved
PPSMOID	word	Optional – PPS MO ID indicating the current version of the PerProviderSubscription MO provisioned to the mobile device. APs do not include this field when the HS2.0 Indication element is used in Beacon or Probe response frames.
ANQPDomainID	word	Optional – ANQP Domain ID indicating the ANQP Domain ID of the AP. ANQP Domain ID value is 0 signifies that the AP has unique ANQP information in one or more of its ANQP elements or Hotspot 2.0 vendor-specific ANQP elements, or has not been implemented with means of knowing whether its ANQP information is unique.

6.6 ANDSF Data Type Definitions

6.6.1 PreferredSSID

Definition PreferredSSID
This prototype defines a structure which describes a preferred network identifier for WLAN access network selection.

Field Name	Type	Description
pSSID	UTF8*	Optional - preferred SSID for WLAN access network selection as defined by [IEEE 802.11-2012]
pHESSID	UTF8*	Optional - preferred HESSID for WLAN access network selection as defined by [IEEE 802.11-2012]

6.6.2 PreferredSSIDs

Definition PreferredSSIDs
This prototype defines a structure which describes the list of preferred SSIDs.

Field Name	Type	Description
numberOfPSSID	dword	The number of Preferred SSID in the array
pPreferredSSIDList	PreferredSSID*	The list of the Preferred SSID

6.6.3 SelectionCriterions

Definition SelectionCriterions
This prototype defines a structure which describes the selection criterions for a dedicated ANDSF policy as defined in [3GPP TS 24.312].

Field Name	Type	Description
criteriaPriority	byte	Priority of the Selection Criterion In case more than one selection criterion exists in a valid WLANSP rule, the UE shall treat the selection criterion with the lowest CriteriaPriority value as the selection criterion having the highest priority among the selection criteria in the WLANSP rule.
pPreferredRoamingPartnerList	PreferredRoamingPartners*	Optional – Roaming partner list. Any roaming partner not in this list has a default priority value of 128 (mid range).
pMinBackhaulThreshold	BackhaulThreshold*	Identical to MinBackhaulThreshold for HS 2.0
maximumBSSLoadvalue	dword	Optional – Identical as Maximum BSS Load value for HS2.0
pRequiredProtoportTuple	ProtoPortTuple*	Optional - Identical as Required Protoport Tuple for HS2.0
pSPExclusionList	UTF8*	Optional - Identical as SP Exclusion List for HS2.0
pPreferredSSIDs	PreferredSSIDs*	Optional – Preferred WLAN access network identifiers. A WLAN access network matches this field if the identifier of the WLAN access network is present in the list then the WLAN access network matches these criteria . If the field is not present or is present and empty, the UE will not evaluate the rule associated.

6.6.4 WLANSP

Definition WLANSP
This prototype defines a structure which describes a WLAN SP as defined in [3GPP TS 24.312].

Field Name	Type	Description
rulePriority	byte	Priority given to one particular rule and is represented as a numerical value. In case more than one valid WLANSP rule exists, the UE shall treat the rule with the lowest RulePriority value as the rule having the highest priority among the valid rules. If the UE finds multiple rules with the same priority, the choice of the rule is UE implementation specific. If there are no matching WLAN access networks according to any rule with a certain priority, the UE may use rules with lower priority.
pSelectionCriteria	SelectionCriterions*	Selection criteria for WLAN access networks.
pValidityArea	UELoc*	Optional – location conditions for a particular rule.
roaming	boolean	Optional – roaming condition for the WLANSP rule. • 0 – Indicates that the rule is only valid when the UE is not roaming. • 1 – Indicates that the rule is only valid when the UE is roaming. The UE shall consider a rule with the Roaming field present as valid only if the current roaming state (roaming/not roaming) of the UE matches the one indicated in the Roaming value and the rule is provided by the H-ANDSF.
pPLMN	UTF8*	PLMN code of the operator, which created this police as defined by [3GPP TS 23.003].
pTimeOfDay	TimeRulesList*	Optional – day condition for a particular flow distribution rule.
updatePolicy	boolean	Optional – the update policy for the WLANSP. • 0 – Indicates that the UE is not required to request an update of the rules. • 1 – Indicates that the UE is required to request an update of the rules. The UpdatePolicy value may be used by the UE to determine whether or not to request an update of its WLANSP when the rule is no longer considered to be valid by the UE. The default value 0 applies if this field is not provisioned.

6.6.5 WLANSPList

Definition WLANSPList
This prototype defines a structure which describes list of WLAN SP.

Field Name	Type	Description
numberOfWLANSP	dword	The number of WLANSP in the array
pWLANSPL	WLANSP*	The list of the WLAN SP

6.6.6 VPLMNPR

Definition VPLMNPR
This prototype defines a structure which describes a list of for one or more VPLMNs with ISMP, ISRP or WLANSP rules which should be preferred for a roaming UE as defined in [3GPP TS 24.312].

Field Name	Type	Description
numberOfVPLMN	dword	The number of VPLMN with preferred rules in the array
pVPLMNL	UTF8*	The list of VPLMN - PLMN codes (MCCMNC) as defined by [3GPP TS 23.003].

6.6.7 RSI

Definition RSI
This prototype defines a structure which describes a Rule Selection Information as defined in [3GPP TS 24.312].

Field Name	Type	Description
pVPLMN	VPLMNPR*	List of VPLMN (visited PLMN) with preferred rules
pPLMN	UTF8*	PLMN code (MCC MNC) of the operator, which created this policy, as defined by [3GPP TS 23.003]. When evaluating the PLMN field the following applies: • if the value contained in this field is equal to the HPLMN (or an equivalent HPLMN) of the UE, the Rule Selection Information is valid • If the value contained in this field is not equal to HPLMN (or any equivalent HPLMN) of the UE, the Rule Selection Information is ignored.

6.6.8 UEProfile

Definition UEProfile
This prototype defines a structure which describes a UE profile as defined in [3GPP TS 24.312].

Field Name	Type	Description
pOSiD	UTF8*	Operating System identifier of the UE: Universally Unique IDentifier (UUID) as specified in [RFC 4122].
pDevCapability	UTF8*	Optional – Device capability of the UE and ANDSF can use this information to adapt the ANDSF MO information to the capabilities of the UE. The value of this field is a sequence of '1' and '0' characters, where the character value '1' denotes support and '0' denotes no support for a feature. The first character of the field value indicates support for ISRP rules based on Application Identifiers (DIDA). If the UE supports ISRP rules based on Application Identifiers, the UE sets the first character of the field value to '1'. The second character of the field value indicates support for IARP rules based on Application Identifiers (DIDA). If the UE supports IARP rules based on Application Identifiers, the UE sets the second character of the field value to '1'. If this field is not present or the value is null string, the UE does not support any feature indicated with the field. If there are additional characters in the field value and the ANDSF server does not understand them, these additional characters are ignored by the ANDSF server. If a character is not '1', the ANDSF server treats the value as '0'.

6.6.9 HomeOperatorPreferences

Definition HomeOperatorPreferences
This prototype defines a structure which describes a Home Operator Preferences as defined in [3GPP TS 24.312].

Field Name	Type	Description
rulePriority	word	Priority given to one particular rule and is represented as a numerical value. In case more than one valid HomeOperatorPreference rule exists, the rule with the lowest RulePriority value as the rule having the highest priority among the valid rules.
S2aConnectivityPreferences	Boolean	Optional - S2a connectivity preference of the home operator:

		• 0 – Indicates that the home operator does not prefer the UE to establish PDN connections over WLAN by using the applicable S2a procedures specified in [3GPP TS 23.402]. • 1 – Indicates that the home operator prefers the UE to establish PDN connections over WLAN by using the applicable S2a procedures specified in [3GPP TS 23.402]. The default value 0 applies if this field is not provisioned.
pValidityArea	UELoc*	Optional - location conditions for a particular HomeOperatorPreference rule
roaming	Boolean	Optional – Roaming condition for a HomeOperatorPreference rule: • 0 – Indicates that the rule is only valid for roaming UE. • 1 – Indicates that the rule is only valid for non-roaming UE.
pPLMN	UTF8*	PLMN code of the operator, which created this policy as defined by [3GPP TS 23.003]. When evaluating the PLMN field the following applies: • if the value contained in this field is equal to the HPLMN (or an equivalent HPLMN) of the UE, the HomeOperatorPreference rule is valid; • If the value contained in this field is neither HPLMN nor an equivalent HPLMN, then the HomeOperatorPreference rule is ignored.
pTimeofDay	TimeRules*	Optional – Day condition for a particular HomeOperatorPreference rule.
updatePolicy	Boolean	Optional – update policy for the HomeOperatorPreference: • 0 – Indicates that the UE is not required to request an update of the rules. • 1 – Indicates that the UE is required to request an update of the rules.

6.6.10 PrioritizedAccess

Definition PrioritizedAccess
This prototype defines a structure which describes a Prioritized Access as defined in [3GPP TS 24.312].

Field Name	Type	Description
accessTechnology	word	The prioritized access technology: • 0 – Reserved

		• 1 – 3GPP • 2 – Reserved • 3 – WLAN • 4 – WiMAX • 5-255 – Reserved
pAccessID	UTF8*	Optional – Identifier for a specific radio access network. Only SSID for WLAN and NAP-ID for WiMAX radio access network are included in this field. The absence of this field indicates that the UE can consider any available radio access network of the defined access technology in the corresponding AccessTechnology field for the network selection.
pSecondaryAccessID	UTF8*	Optional – Identifier for a specific radio access network. Only HESSID for WLAN radio access network is contained in this filed as defined by [IEEE 802.11-2012] The SecondaryAccessID field may only be present when the corresponding AccessID field for a WLAN radio access network is present.
accessNetworkPriority	byte	In case more than one valid PrioritizedAccess are available and if the value of the priority belongs to the range 1-250, the UE shall consider the access network (with the corresponding access identifier if present) with the lowest AccessNetworkPriority value as the access network (with the corresponding access identifier if present) having the highest priority. The AccessNetworkPriority value 'Restricted access' (254) indicates an access that should not be used by the UE. The AccessNetworkPriority value 'Forbidden' (255) indicates an access that shall not be used by the UE. The same AccessNetworkPriority value may be used for more than one AccessID and more than one Access Technology. If more than one AccessID or more than one Access Technology with the same value of the AccessNetworkPriority are available, the UE selects one of them in an implementation dependant way. If the UE is not able to find an access network according to ANDSF policies, it is implementation dependent how to proceed with network selection • 0 – Reserved • 1-250 – Priority value • 251-253 – Reserved • 254 – Restricted access. This access should be avoided if the current rule is active. • 255 – Forbidden. UE is not allowed to use this access if the current rule is active.

6.6.11 3GPPLoc

Definition 3GPPLoc
This prototype defines a structure which describes a 3GPP location as defined in [3GPP TS 24.312].

Field Name	Type	Description
pPLMN	UTF8*	PLMN code for one particular 3GPP location condition as defined by [3GPP TS 23.003].
pTAC	UTF8*	Optional – Tracking Area Code for one particular 3GPP location condition as defined by [3GPP TS 23.003].
pLAC	UTF8*	Optional – indicates a Location Area Code for one particular 3GPP location condition as defined by [3GPP TS 23.003].
GERAN_CI	word	Optional – The format of the Cell Global Identity, of which the Cell Identity is part, as defined by [3GPP TS 23.003].
UTRAN_CI	bit[28]	Optional – The UTRAN_CI value is set to the UTRAN Cell Identity as defined in [3GPP TS 25.331].
EUTRA_CI	bit[28]	Optional – The EUTRA_CI value is set to the cell identity part of the Evolved Cell Global Identifier, as described in [3GPP TS 36.331].

6.6.12 1xElements

Definition 1xElements
This prototype defines a structure which describes elements related to 1x in a 3GPP2 location as defined in [3GPP TS 24.312].

Field Name	Type	Description
pSID	UTF8*	System Identification code for one particular 3GPP2 1x RAT location condition for the ISMP rule as defined by [3GPP2 C.S0016]
pNID	UTF8*	Optional – Network Identification code for one particular 3GPP2 1x RAT location condition for the ISMP rule as defined by [3GPP2 C.S0016]
pBaseID	UTF8*	Optional - Base Station Identification code for one particular 3GPP2 1x RAT location rule as defined by [3GPP2 C.S0005]

6.6.13 HRPDElements

Definition HRPDElements
This prototype defines a structure which describes elements related to HRPD in a 3GPP2 location as defined in [3GPP TS 24.312].

Field Name	Type	Description
Sector_ID	qword	Sector Identification code for one particular 3GPP2 HRPD RAT location as defined by [3GPP2 C.S0024-0].
netmask	byte	Corresponding Netmask code for one particular 3GPP2 HRPD RAT location as defined by [3GPP2 C.S0024-0].

6.6.14 3GPP2Loc

Definition 3GPP2Loc
This prototype defines a structure which describes a 3GPP2 location as defined in [3GPP TS 24.312].

Field Name	Type	Description
p1x	1xElements*	Optional – 3GPP2 1x RAT location If the UE is currently aware that it is located in the coverage area described by at least one instance of fields of 1x, the UE shall consider the corresponding rule as valid. If the location is indicated with more than one subfields (i.e. SID and at least one subfield out of NID or Base_ID are present) within a single instance of 1x field, the UE shall consider 3GPP2 location validity condition for the particular rule to be fulfilled only if all the present values in the 1x field match with the location of the UE.
pHRPD	HRPDElements*	Optional – 3GPP2 HRPD RAT location descriptions. If the UE is currently aware that it is located in the coverage area described by at least one instance of the HRPD field, the UE shall consider the corresponding rule as valid. The UE shall consider 3GPP2 location validity condition for the particular rule to be fulfilled only if both the subfields values within a single instance of the HRPD field (i.e. Sector_ID and Netmask) match with the location of the UE.

6.6.15 WLANLoc

Definition WLANLoc
This prototype defines a structure which describes a WLAN location as defined in [3GPP TS 24.312].

Field Name	Type	Description
pHESSID	UTF8*	Optional - The HESSID for a particular WLAN location as defined by [IEEE 802.11-2012]
pSSID	UTF8*	Optional – SSID for a particular WLAN location as defined by [IEEE 802.11-2012]
pBSSID	UTF8*	Optional – BSSID – the AP identifier for one particular WLAN location as defined by [IEEE 802.11-2012]

6.6.16 GeoLocations

Definition GeoLocations
This prototype defines a structure which describes locations areas through circular elements as defined in [3GPP TS 24.312].

Field Name	Type	Description
numberOfGeoC	dword	The number of Geo Locations in the array
pCircularL	GeoCircular*	List of Circular locations

6.6.17 GeoCircular

Definition GeoCircular
This prototype defines a structure which describes a circular location in terms of latitude, longitude and radius.

Field Name	Type	Description
pAnchorLatitude	UTF8*	Optional – Latitude value of the centre of the circular area as defined in [3GPP TS 23.032].
pAnchorLongitude	UTF8*	Optional – Longitude value of the centre of the circular area as defined in [3GPP TS 23.032].
pRadius	UTF8*	Optional - Radius value of the circular area, given in meters as defined in [3GPP TS 23.032].

6.6.18 UELoc

Definition UELoc
This prototype defines a structure which describes a UE location element as defined in [3GPP TS 24.312].

Field Name	Type	Description
p3GPPLocation	3GPPLoc*	If the location is indicated with more than one field (i.e. PLMN and at least one subfield out of TAC, LAC, GERAN_CI, UTRAN_CI or EUTRA_CI) within a single instance of 3GPPLocation, then the UE shall consider 3GPP location validity condition for the particular rule to be fulfilled only if the present field values in the field match with the location of the UE in that radio access technology, and the UE shall ignore other existing values, if any.
p3GPP2Location	3GPP2Loc*	If the UE is currently aware that it is located in the coverage area described by the fields 1x or HRPD of 3GPP2Location or both, the UE shall consider the corresponding rule as valid. In case of overlapping validity domains of multiple policy rules, RulePriority field is used as discriminator. If ANDSF provides 3GPP2Location as a validity area, either the 1x or HRPD fields, or both, shall be provided.
WiMaxLocation	dword	Optional Reserved for future use – not supported by OpenCMAPI 1.1
pWLANLocation	WLANLoc*	Optional – Location of WLAN If the UE is currently aware that it is located in the coverage area described by at least one instance of WLANLocation, the UE shall consider the corresponding rule as valid. In case of overlapping validity domains of multiple policy rules, RulePriority field is used as discriminator. The UE shall ignore WLANLocation field that is present and does not contain at least one of the non-empty leaves (i.e. HESSID, SSID or BSSID). If the location is indicated with at least one present subfield out of HESSID, SSID, or BSSID within a single instance of WLANLocation, the UE shall consider WLAN location validity condition for the particular rule to be fulfilled only if all the present subfield values in the field match with the location of the UE.
pGeoLocation	GeoLocations*	Optional – Geo locations of possible areas If the UE is currently aware that it is located in the area described by at least one instance of the field, the UE shall consider the corresponding rule as valid. In case of overlapping validity domains of multiple policy rules,

		RulePriority field is used as discriminator. If the GeoLocation is present and empty, then the field is not considered when evaluating the validity of the corresponding rule. The UE shall consider Geo location validity condition for the particular rule to be fulfilled only if all the subfield values within a single instance of GeoCircular (i.e. AnchorLatitude, AnchorLongitude and Radius) match with the location of the UE.
pRPLMN	UTF8*	Optional – a PLMN code of the registered PLMN the UE is connected to as defined by [3GPP TS 23.003].

6.6.19 TimeRules

Definition TimeRules
This prototype defines a structure which describes rules associated to time and day as defined in [3GPP TS 24.312].

Field Name	Type	Description
pTimeStart	UTF8*	Optional – Time of day the rule starts. Formatted as defined in ISO 8601
pTimeStop	UTF8*	Optional – Time of day the rule stops. Formatted as defined in ISO 8601
pDateStart	UTF8*	Optional – Date the rule start Formatted as defined in ISO 8601: YYYY-MM-DD HH:MM:SS If DateStart indicates a date that does not exist is, the value will be considered as not existing.
pDateStop	UTF8*	Optional – Date the rule stop Formatted as defined in ISO 8601: YYYY-MM-DD HH:MM:SS If DateStop indicates a date that does not exist is, the value will be considered as not existing.
dayofWeek	byte	Optional – day(s) of the week for which the corresponding rule is valid. The value is formatted as a bit map representing days of the week. The most significant bit is set to one. The remaining bits represent days of the week (Bit 1: Monday, ..., Bit 7: Sunday) If the bit corresponding to a day of the week is set, the rule is valid on that day. The following values are invalid:

		• "null"; • 128; and • less than 128. If an invalid value is provided, the value will be considered as not existing.

6.6.20 TimeRulesList

Definition TimeRulesList
This prototype defines a structure which describes a list of different time lists.

Field Name	Type	Description
numberOfTR	dword	The number of Time Rules in the array
pTimeRulesl	TimeRules*	List of the time rules

6.6.21 ANDSFPolicy

Definition ANDSFPolicy
This prototype defines a structure which describes a subscription policy as provided through an ANDSF MO as defined in [3GPP TS 24.312].

Field Name	Type	Description
rulePriority	word	In case more than one valid ISMP rule exists, the UE shall treat the rule with the lowest RulePriority value as the rule having the highest priority among the valid rules. If the UE finds multiple rules with the same priority, the choice of the rule is UE implementation specific. If there are no matching access networks according to the rule, the UE shall use other rules with the same priority. If there are no matching access networks according to any rule with a certain priority, the UE may use rules with lower priority.
pPreferredAccess	PrioritizedAccess*	The preferred access for one particular policy rule.
pValidityArea	UELoc*	Optional – The validity condition of ValidityArea is considered valid when at least one of 3GPPLocation, or 3GPP2Location or WiMAXLocation, or WLANLocation, or GeoLocation is a match. If the ValidityArea field is present and empty (i.e. none of the fields 3GPPLocation, 3GPP2Location, WiMAXLocation, WLANLocation or GeoLocation exist), then the ValidityArea is not considered when evaluating the validity of the corresponding rule.

		If the UE cannot deduce its location by any means, only rules without Validity Area field or rules with empty ValidityArea field can be considered for valid rule.
roaming	boolean	Roaming Condition: • 0 – Indicates that the rule is only valid when the UE is not roaming. • 1 – Indicates that the rule is only valid when the UE is roaming.
pPLMN	UTF8*	PLMN code of the operator, which created this policy. The format of the PLMN is defined by 3GPP TS 23.003.
pTimeofDay	TimeRules*	Optional – The UE shall interpret the TimeOfDay field values as related to the local time of the UE.
updatePolicy	boolean	Optional – Indication of update policy for the ISMP. • 0 – Indicates that the UE is not required to request an update of the ISMP. • 1 – Indicates that the UE is required to request an update of the ISMP. The UpdatePolicy value may be used by the UE to determine whether or not to request an update of its ISMP when the rule is no longer considered to be valid by the UE. The default value 0 applies if this field is not provisioned.

6.6.22 DiscoveryInfo

Definition DiscoveryInfo
This prototype defines a structure which describes a discovery information element as defined in [3GPP TS 24.312].

Field Name	Type	Description
accessNetworkType	word	Type of the Access network for which discovery assistance information is provided: • 0 – Reserved • 1 – 3GPP • 2 – Reserved • 3 – WLAN • 4 – WiMAX • 5-255 – Reserved
pAccessNetworkArea	UELoc*	Description of the location where one particular access network is expected to be available. If the UE is aware that it is located in at least one of

		the locations described by this field, the UE may assume the corresponding access network to be available. If the AccessNetworkArea field is present and empty (i.e. none of the fields 3GPPLocation, 3GPP2Location, WiMAXLocation, WLANLocation or GeoLocation exist), then the AccessNetworkArea is not considered when evaluating the discovery information.
pAccessNetworkInfoRef	UTF8*	Optional – Reference to an object with access network type specific information.
pPLMN	UTF8*	PLMN code of the operator which created this discovery information as defined by [3GPP TS 23.003].

6.6.23 ISRPList

Definition ISRPList
This prototype defines a structure which describes a list of ISRP. Note: ISRP is not supported by OpenCMAPI 1.1 – this structure is a place holder for future versions of OpenCMAPI. The structure is considered as variable

6.6.24 IARPList

Definition IARPList
This prototype defines a structure which describes a list of IARP Note: IARP is not supported by OpenCMAPI 1.1 – this structure is a place holder for future versions of OpenCMAPI. The structure is considered as variable

6.6.25 ANDSFSubscription

Definition ANDSFSubscription
This prototype defines a structure which describes a subscription profile (subscription policy, discovery information, location, ISRP, UE profile...) for ANDSF as defined in [3GPP TS 24.312].

Field Name	Type	Description
updateIdentifier	word	Identifies if there is existing provisioned MO subscription or not

		• 0x0000: MO not present (default) • Other: Existing provisioned ANDSF MO
pName	UTF8*	Optional – User displayable name of the ANDSF MO for the ANDSF MO settings
pPolicy	ANDSFPolicy*	Optional – ANDSF policy.
pDiscoveryInformation	DiscoveryInfo*	Optional – Access network discovery information. An operator may provide information on available access networks through the ANDSF. The UE may use the information as an aid in discovering other access networks.
pUELocation	UELoc*	Optional – Current location of the UE.
pISRP	ISRPList*	Optional – policies for ISRP ISRP is not supported by OpenCMAPI 1.1 – this field is ignored for the time being
pUEProfileInfo	UEProfile*	Optional – information characterizing the UE
pWLANSP	WLANSPList*	Optional – WLANSP information (a set of one or more WLANSP rules) At any point in time, there shall be at most one rule applied, that rule is referred to as the 'active' rule. There can be multiple valid WLANSP rules at the same time.
pIARP	IARPList*	Optional – policies for IARP IARP is not supported by OpenCMAPI 1.1 – this field is ignored for the time being
pRuleSelectionInformation	RSI*	Optional – rule selection information indicating VPLMNs with preferred WLAN selection rules.
pHomeOperatorPreference	HomeOperatorPreferences*	Optional – Home Operator Preference information

6.7 Callback Data Type Definitions

6.7.1 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

6.7.2 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 • 0x00000025: Incoming Voice Call • 0x00000026: SEServicesChange • 0x00000027: Battery Status Changed • 0x00000028: Battery Threshold reached • 0x00000029: WLAN Settings Change • 0x0000002A: WLAN New MO • 0x0000002B: Get Mobility to Location Complete • 0x0000002C: P2P Discovery Match • 0x0000002D: P2P Connection established • 0x0000002E: P2P Group Notification

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.

7.2.4 CMAPI_API_GetFunctionsSupported()

The **CMAPI_API_GetFunctionsSupported()** function retrieves the OpenCMAPI groups of functions supported by the enabler.

Prototype
dword **CMAPI_API_GetFunctionsSupported** (CMAPIFunction* pFunctionsSupported)

Parameters		
Field Name	**Mode**	**Description**
pFunctionsSupported	Output	Pointer to get the OpenCMAPI functions supported by the enabler in bitmap in accordance with CMAPIFunction definition.

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 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* pServingNetworkInfoSize, 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.
pServingNetworkInfoSize	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 pServingNetworkInfoSize 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	The 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 Cellular 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 names.

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 a 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 (enable or disable) 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	The 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	The 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 or 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 or 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 or 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 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.

7.7.20 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 tunnelling 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 tunnelling 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 behaviour 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 behaviour 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.

7.8.20 CMAPI_DevSrv_DevAttributes()

The **CMAPI_DevSrv_DevAttributes()** function is used to provide to application information regarding device attributes (e.g. screen, keypad, camera, microphone, loudspeaker).

Prototype

dword **CMAPI_DevSrv_DevAttributes** (dword deviceID, DevAttributes* pDevAttributes, dword* pDevAttributesSize)

Parameters

Field Name	Mode	Description

deviceID	Input	The ID of the device concerned
pDevAttributes	Output	Capabilities supported by the device. See DevAttributes definition
pDevAttributesSize	Input/Output	The size of pDevAttributesSize in bytes. 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
0X30000029	The pDevAttributes 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.

7.9 Device Extended Service APIs

This section defines a set of APIs to address extended services a device can support. In the version 1.1 of OpenCMAPI enabler, two sets of functionalities have been identified.

- NFC (Near Field Communication)
- SE (Secure Element)

The main goal is to expose to an application (connection or non connection application manager) what extended functionalities a device can support and not to provide a full set of APIs unless they are not specified by other standard development organizations.

Applications may need to know which services are available and enabled by the device. The OpenCMAPI enabler and its set of API functionalities is a means to discover which services are or can be offered by a device. Therefore an application can be able to select a service among a list provided by the device through OMA APIs or other APIs from e.g. SIM Alliance and Global Platform.

Some class of services are easily identified by their AID (Application ID). Some of those AID are public and well known (e.g. Telecom, Payment). Many services exist and will exist. Those services will need security and will use mechanisms based on secure elements embedded in the device.

In addition, this function can be used to expose services which don't need sophisticated security mechanisms.

7.9.1 CMAPI_ExtDevSrv_NFC()

The **CMAPI_ExtDevSrv_NFC ()** function is used to provide to application information regarding NFC functionalities available in the device.

Prototype
dword **CMAPI_ExtDevSrv_NFC** (dword deviceID, NFCMode* NFC)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
NFC	Output	All NFC capabilities supported by the device

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.

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.

7.9.2 CMAPI_ExtDevSrv_SE()

The **CMAPI_ExtDevSrv_SE()** function is used to provide to application information regarding SE (Secure Element) functionalities and services available in the device.

Prototype
dword **CMAPI_ExtDevSrv_SE** (dword deviceID, SEServices* pSEServices, dword* pSEServicesSize)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pSEServices	Output	All SE services supported by the device. See SEServices structure definition
SEServicesSize	Output/Input	The size of pSEServicesSize in bytes. Will contain the minimum bytes 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.
0X30010004	pSEServices is not large enough, pSEServicesSize 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.

7.10 PINs/PUKs Management APIs

7.10.1 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.

7.10.2 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.

7.10.3 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.

7.10.4 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.

7.10.5 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.

7.10.6 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.

7.10.7 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.

7.11 UICC Management APIs

7.11.1 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.

7.11.2 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] and [3GPP TS 51.014] 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.

7.11.3 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] and [3GPP TS 51.014] 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.

7.11.4 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] and [3GPP TS 51.014] 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.

7.11.5 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] and [3GPP TS 51.014] 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.

7.12 WLAN APIs

7.12.1 WLAN connection introduction

With the introduction of Hotspot 2.0 ([Wi-Fi Alliance HS2.0 TS] also known as Passpoint), Wi-Fi Access Points are offering more information to the terminal and are allowing it to automatically connect to them in a secure and easy way.

In addition, 3GPP ANDSF (Access Network Discovery and Selection Function [3GPP TS 24.312]) purpose is to assist the terminal to discover non-3GPP access networks – such as WLAN Networks – that can be used in addition to 3GPP access networks and to provide the terminal with rules defining policies related to connection to these networks.

Both of these technologies are providing different rules and policies attached to them through Management Objects that can be provided to the SIM/R-UIM/NAA on UICC or to the Management Object tree of the terminal if the terminal supports OMA Device Management MO protocol.

When both technologies (MO) are present in the terminal, the user and operator settings will be used to determine which technologies is preferred and which WLAN operator subscription should be used. The user settings has priority over operator settings.

If ANDSF (resp. HS2.0) is preferred, then selection on networks to be connected to will be done accordingly to the policies present within the ANDSF MO (resp. HS2.0 MO) and then, if no connection was found meeting the criteria of these policies, accordingly to the policies present within the HS2.0 MO (resp. ANDSF MO). In case of several subscriptions for either HS2.0 or ANDSF, the respective default subscriptions can be found in the user settings.

If ANDSF only is selected, then selection on networks to be connected to will be done accordingly to the policies present only within the ANDSF MO and the policies (including restricted networks) present within the HS2.0 MO should not be considered at all.

If no preferences is specified then both policies of ANDSF and HS2.0 are used to select network. In case of networks having same priority value in both lists, the network within ANDSF list has priority on the network on HS2.0 list.

7.12.2 WLAN Network Lists Management

7.12.2.1 Known and Unknown WLAN Networks

A Known WLAN network is a Network (SSID, networks identifiers) that has already been used or predefined by the user or prelisted by the operator within the terminal or within the SIM/R-UIM/NAA on UICC or through any operator policy (MO).

If the network is Known, it will belong to the Known Network List.

All other WLAN networks (for example new network not already listed) are considered as Unknown.

7.12.2.2 User Preferences Lists

An user can manage Known networks and organized them in 2 lists (exclusive of each other):

- User Preferred Network List: List of the WLAN Networks the user would like to have automatic connection to or that are preferred to use (for example, Home WLAN) – corresponding to the User Controlled WLAN specific Identifier List in 3GPP (see [3GPP TS 24.234] and [3GPP TS 31.102]) – Each network belonging to this list has a priority affected by the user (by default, the priority value of a network in the User Preferred Network List is 0 - unknown)).
- BlackList: WLAN Network the user has decided it cannot be connected to – Manually or Automatically.

7.12.2.3 Operator Preferences Lists

Though the policies enabled in the SIM/R-UIM/NAA on UICC or in the Management Object tree of the device if the Device supports OMA Device Management MO protocol (i.e. Hotspot 2.0 MO [Wi-Fi Alliance HS2.0 TS], ANDSF MO [3GPP TS 24.312]), an operator or Home Service Provider can provide 2 different lists related to WLAN networks:

- Operator Preferred Network List: List of the WLAN Networks preferred by the operator and for which automatic connection can be done (SSID of the Home Service provider, roaming partners) – corresponding to the Operator

Controlled WLAN specific Identifier List in 3GPP (see [3GPP TS 24.234] and [3GPP TS 31.102] – Each network belonging to this list has a priority affected by the operator (by default, the priority value of a network in the User Preferred Network List is 0 – Unknown). A Network listed in the Operator Preferred Network List cannot belong to the Blacklist.

- Restricted or Excluded Network List: List of SSIDs not preferred by the Operator or Home Service Provider. A network belonging to this list cannot be connected automatically to except if it belongs to the User Preferred Network List and it can be selected manually by the user.

Any network belonging to one of the lists is considered as Known and belongs to the Known Network List.

7.12.2.4 List management and priorities

The user preference list has higher priority than the operator preference list.

The fact that a network belongs to one of the preference lists (either user or operator) has impacts and is influencing the connection:

- In Manual Connection mode, all networks with the exception of the networks belonging to the Blacklist can be connected to. If there is a need to establish a connection to a network belonging to the Blacklist, this network needs first to be removed from the blacklist.
- In Automatic Mode, the User Preferred Network List has priority over the Operator Preferred Network List.
    - When networks are discovered, the OpenCMAPI will compare them with the User Preferred Network List and, if several matches are found, will connect (associate) with the WLAN Network having the highest priority in this list.

If no match was found within the User Preferred Network List then the Operator Preferred Network List and the Excluded Network List as well as priorities and rules defined in [3GPP TS 31.102], [3GPP TS 24.234], [3GPP TS 23.234] will be used to evaluate possible match. If a network belongs to the Excluded Network List, automatic connection to it will not be possible. The discovered network list is filtered based on implementation dependent criteria (minimum RSSI level…) and matching credentials then accordingly to the Operator Preferred Network List, and, if several matches are found, will associate with the WLAN Network having the highest priority in this list.

7.12.3 Automatic/Manual Connection to WLAN

7.12.3.1 Connection Modes

Two modes of selection are existing for connection to WLAN:

- Automatic Network Selection (default mode)
- Manual Network Selection

Both are based first on the user preferences then on the operator preferences.

7.12.3.2 Manual Mode

In manual mode, the device cannot automatically connect to an Access Point. It needs to be connected manually.

Two options are possible:

- Manual connection to all networks (known and unknown networks) – it is the general case.
- Manual connection only to known networks (this is subject to dedicated policies – For example a corporate allowing only to connect to a list of dedicated networks)

7.12.3.3 Automatic Mode

In Automatic Mode, the device can only connect to a known Network – the device is associated and authenticated automatically (without user interaction) to a WLAN Network based on the priorities defined above.

The Network selection process in Automatic Mode follows the algorithms below.

First, the user preferences including user preferred network list are used to associate and connect automatically if there is a match.

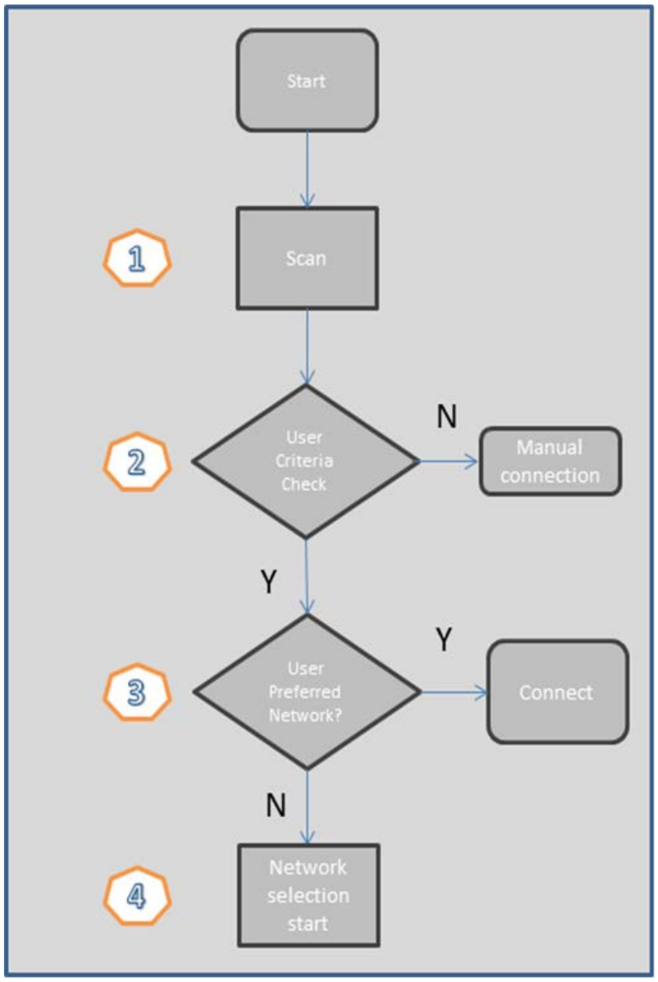

Figure 1: WLAN Automatic Connection – User preferences

The following table is listing the steps corresponding to the WLAN Automatic Connection part 1.

WLAN Automatic Connection	
Steps	**Description**
1	On start-up, the CM Application initiates a scan through the CMAPI_WLAN_Scan_Async() function and the result is reported in callback CMAPI_Callback_ScanWLANComplete().
2	The CM Application proceeds to criteria check for example mobility. If the criteria are met, the process can proceed further.
3	Comparison between the discovered networks with User preferred networks list. If a match is found, the OpenCMAPI authenticates and associates directly with WLAN AP having the highest user priority (the relevant authentication sequence is done with the highest priority to SIM/USIM based EAP methods).

4	The Network Selection process can start

Table 6: Steps related to WLAN Automatic Connection – User Preferences

If there is no match in the User Preferred List, then the Network selection process in Automatic Mode follows the algorithm below:

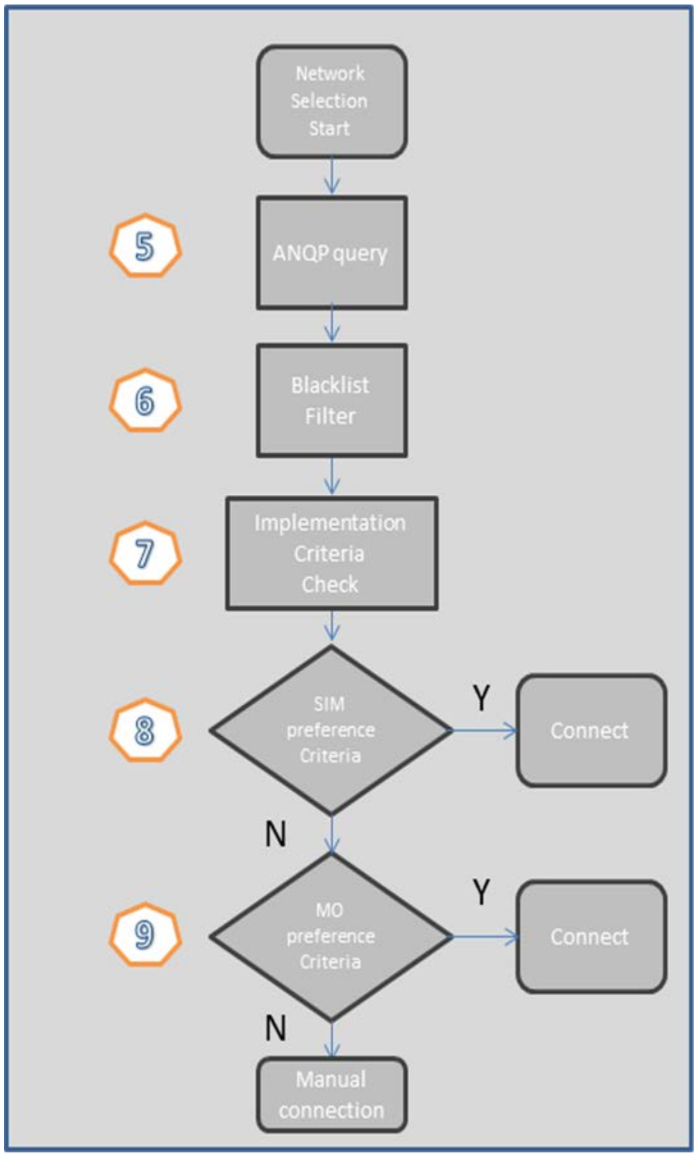

Figure 2: WLAN Automatic Connection - Network Selection

The following table is listing the steps corresponding to the Network Selection:

WLAN Automatic Connection	
Steps	**Description**
5	CM Application proceeds to querying the ANQP information of all HS2.0 networks

	identified. In particular, the CM Application can obtain the 3GPP/3GPP2 Cellular Information provided by the WLAN.
6	The AP corresponding to a blacklisted network are filtered out
7	The AP not matching implementation dependent criteria such as the minimum RSSI level are filtered out.
8	The remaining discovered networks are compared with the preferred list from the SIM/USIM/R-UIM/NAA on UICC. If a match is found, the OpenCMAPI authenticates and associates directly with WLAN AP having the highest priority (the relevant authentication sequence is done with the highest priority to SIM/USIM based EAP methods). If a match is found but the network with the highest priority was never connected to and the related user setting is set to “No Automatic Connection to WLAN if it is first time connection to the WLAN Network” then the CM Application shall be prompted to a manual connection to this network.
9	Then, the remaining discovered networks are filtered for realms matching the user’s credentials. APs whose NAI Realm, 3GPP Cellular Network and OIs do not match user credentials (optionally including credential types) are excluded. If a match is found, the OpenCMAPI authenticates and associates directly with WLAN AP having the highest priority (the relevant authentication sequence is done with the highest priority to SIM/USIM based EAP methods). If a match is found but the network with the highest priority was never connected to and the related user setting is set to “No Automatic Connection to WLAN if it is first time connection to the WLAN Network” then the CM Application shall be prompted to a manual connection to this network.

Table 7: Steps related to WLAN Automatic Connection – Network Selection

7.12.4 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.

7.12.5 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.

7.12.6 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.

7.12.7 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.

7.12.8 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.

7.12.9 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.

7.12.10 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.

7.12.11 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
0X00012004	Authentication failure
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.
0XF0000006	The WLAN Encryption Type used is not allowed. Please use proper Encryption type

7.12.12 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
0X00012004	Authentication failure
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.
0XF0000006	The WLAN Encryption Type used is not allowed. Please use proper Encryption type

7.12.13 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.

7.12.14 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. • 0x00000000: Auto connect to known networks(default value) • 0x00000001: Auto connect only to networks for which the option automatic mode has been selected

		• 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.

7.12.15 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. • 0x00000000: Auto connect to known networks (default value) • 0x00000001: Auto connect only to networks for which the option automatic mode has been selected • 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.

7.12.16 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.

7.12.17 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.

7.12.18 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.

7.12.19 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.

7.12.20 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.

7.12.21 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.

7.12.22 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.

7.12.23 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.

7.12.24 CMAPI_WLAN_EnableCapability()

The **CMAPI_WLAN_EnableCapability ()** function is used to enable or disable the WLAN feature in the device.

Prototype
dword **CMAPI_WLAN_EnableCapability** (dword deviceID, dword WLANswitch)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The device identifier
WLANswitch	Input	The WLAN feature switch • 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.
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.

7.12.25 CMAPI_WLAN_AuthenticationSupported()

The **CMAPI_WLAN_AuthenticationSupported()** function is used to determine if HS2.0 is supported by the device and what are the authentications methods supported.

Prototype
dword **CMAPI_WLAN_AuthenticationSupported** (dword deviceID, dword* pHS2Support, dword*

```
pHS2version, dword* pAuthentication)
```

Parameters

Field Name	Mode	Description
deviceID	Input	The ID of the device concerned
pHS2Support	Output	Indicates HS2.0 support: • 0x00000000: HS2.0 not Supported • 0x00000001: HS2.0 Supported
pHS2version	Output	Indicates which version of HS is supported: • 0x00000000: if HS2.0 not Supported • 0x00000001: Phase 1 • 0x00000002: Phase 2 • 0x00000003: Phase 3 • 0x00000004: Reserved for future use • 0x00000005: Reserved for future use
pAuthentication	Output	The EAP Authentication methods supported (see Authentication methods) as a bitmap (0x00000000 indicates none): • 0x00000001: MD5-Challenge • 0x00000002: Generic Token Card (GTC) • 0x00000004: EAP-TLS • 0x00000008: LEAP • 0x00000010: EAP-SIM • 0x00000020: EAP-TTLS • 0x00000040: EAP-AKA • 0x00000080: PEAP • 0x00000100: EAP MS-CHAP-V2 • 0x00000200: EAP-FAST • 0x00000400: EAP-PSK • 0x00000800: EAP-IKEv2 • 0x00001000: EAP-AKA’

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.

7.12.26 CMAPI_WLAN_ManageKnownNetwork()

The **CMAPI_WLAN_ManageKnownNetwork()** function is used to add/delete or update a network to or in the known network list.

Prototype
dword **CMAPI_WLAN_ManageKnownNetwork** (dword index, WLANNetwork* pNetwork, dword operation, WLANPreferences* Preferences)

Parameters		
Field Name	**Mode**	**Description**
index	Input	The zero based index which describes the position of the network in the known networks list. In case of addition, any existing subsequent entry will have their previous index adjusted to be one larger. In case of deletion, any subsequent records will have their index decremented.
pNetwork	Input	The network to add/delete or update to the known networks list.
operation	Input	The operation type to Add, Delete, Update the Wlan Network in the known network list: • 0x00000001: Add to the known network List • 0x00000002: Delete from the known network List • 0x00000003: Update an existing known network record This action is as well updating the field Known in the WLANNetwork record.
Preferences	WLANPreferences*	In case of Add or Update, provides the User and Operator preferences. If filled, this overrides the preferences provided with the input of pNetwork. This is in particular updating the User preference and Operator preference lists.

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
0X00010020	The WLAN Network cannot be blacklisted (i.e. already in the operator preferred network list)
0X0001001A	The User Preference is invalid
0X0001001B	The User Preference Priority is invalid
0X0001001C	The Operator Preference is invalid
0X0001001D	The Operator Preference Priority 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.

7.12.27 CMAPI_WLAN_GetANQP()

The **CMAPI_WLAN_GetANQP()** function is used to get the ANQP information including HS2.0 ANQP.

Prototype
dword **CMAPI_WLAN_GetANQP** (dword deviceID, dword dialogToken, APE* AdvertisementProtocolElement, word queryRequestlength, ANQPQueryList* ANQPQL, HSQueryList* HSQL, word responselength, ANQPAnswer p ANQPQResponse, dword* pANQPQResponseSize, HS2ANQPAnswer pHSQResponse, dword* pHSQResponseSize)

Parameters

Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
dialogToken	Input	Used for matching action responses with action requests in case of multiple concurrent actions (it is the responsibility of the CM Application to use different numbers in this case).

AdvertisementProtocolElement	Input	The Advertisement Protocol Element as defined in the structure APE with the Advertisement Protocol ID set to 0 (ANQP).
queryRequestlength	Input	The Query Request length corresponding to the total number of bytes for the Query List ANQP element and the HS Query List.
ANQPQL	Input	The Query List ANQP element
HSQL	Input	The HS Query List
responselength	Output	The length of the response corresponding to the total number of bytes for the ANQP capabilities and responses to the queries.
pANQPQResponse	Output	The response to the Query List ANQP element(s) requested. The length is depending of the elements required (see structures for ANQP)
pANQPQResponseSize	Input/Output	The size of the pANQPQResponse buffer in bytes
pHSQResponse	Output	The response to the HS Query List element(s) requested. The length is depending of the elements required (see structures for ANQP)
pHSQResponseSize	Input/Output	The size of the pHSQResponse 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.
0X00014001	The Advertisement Protocol Element is invalid
0X00014002	The Query List ANQP element is invalid
0X00014003	The HS Query List is invalid
0X3000002A	The pANQPQResponse buffer is not large enough. pANQPQResponseSize contains the minimum buffer length required.
0X3000002B	The pHSQResponse buffer is not large enough. pHSQResponseSize 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.

7.12.28 CMAPI_WLAN_Get_WSIDL()

The **CMAPI_WLAN_Get_WSIDL()** function is used to retrieve the user preferred list and operator preferred list of WLAN specific identifier (WSID) from the SIM/R-UIM/NAA on UICC in accordance with [3GPP TS 31.102].

Prototype
dword **CMAPI_WLAN_Get_WSIDL** (dword deviceID, EF_UWSIDL* pUWSIDL, dword* pUWSIDLSize, EF_OWSIDL* pOWSIDL, dword* pOWSIDLSize)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pUWSIDL	Output	The user preferred list of WLAN specific identifiers (null value if not present)
pUWSIDLSize	Input/Output	The size of the pUWSIDL buffer in bytes
pOWSIDL	Output	The operator preferred list of WLAN specific identifiers (null value if not present)
pOWSIDLSize	Input/Output	The size of the pOWSIDL 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.
0X30040031	The pUWSIDL buffer is not large enough. pUWSIDLSize contains the minimum buffer length required.
0X30040032	The pOWSIDL buffer is not large enough. pOWSIDLSize 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.

7.12.29 CMAPI_WLAN_Get_HS2MOSubcription()

The **CMAPI_WLAN_Get_HS2MOSubcription()** function is used to retrieve the elements related to HS2.0 subscriptions in accordance with [Wi-Fi Alliance HS2.0 TS].

Prototype
dword **CMAPI_WLAN_Get_HS2MOSubcription** (dword deviceID, PerProviderSubscription*

pHS2MOSubscription, dword* pHS2MOSubscriptionSize)

Parameters

Field Name	Mode	Description
deviceID	Input	The ID of the device concerned
pHS2MOSubscription	Output	The detail of the HS2.0 subscriptions present in the device
pHS2MOSubscriptionSize	Input / Output	The size in byte of pHS2MOSubscription.

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.
0X30040021	The pHS2MOSubscription is not large enough.
0X00014001	HS 2.0 MO is not supported by the device
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.

7.12.30 CMAPI_WLAN_Get_ANDSFMOSubcription()

The **CMAPI_WLAN_Get_ANDSFMOSubcription()** function is used to retrieve the elements related to ANDSF subscription in accordance with [3GPP TS 24.312].

Prototype
dword **CMAPI_WLAN_Get_ANDSFMOSubcription** (dword deviceID, ANDSFSubscription* pAMOSubscription, dword* pAMOSubscriptionSize)

Parameters

Field Name	Mode	Description
deviceID	Input	The ID of the device concerned
pAMOSubscription	Output	The detail of the ANDSF MO subscription present in the device
pAMOSubscriptionSize	Input / Output	The size in byte of pAMOSubscription.

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.
0X00014002	ANDSF MO is not supported by the device
0X30040022	The pAMOSubscription 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.

7.12.31 CMAPI_WLAN_Get_WLANSettings()

The **CMAPI_WLAN_Get_WLANSettings()** function is used to retrieve the user and operator settings for WLAN.

Prototype
dword **CMAPI_WLAN_Get_WLANSettings** (dword deviceID, WLANUserSettings* pUserSettings, WLANOperatorSettings* pOperatorSettings)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pUserSettings	Output	The user settings for WLAN
pOperatorSettings	Output	The Operator settings for 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.

7.12.32 CMAPI_WLAN_Set_WLANUserSettings()

The **CMAPI_WLAN_Set_WLANUserSettings()** function is used to set the user settings for WLAN.

Prototype
dword **CMAPI_WLAN_Set_WLANUserSettings** (dword deviceID, dword automaticConnection, AutoConnectionSettings* AutomaticConnectionOptions, UTF8* HS20DefaultSubscription, UTF8* ANDSFDefaultSubscription, dword userSetting)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
automaticConnection	Input	Specifies which type of Connection mode needs to be supported: • 0x00000000: Manual Only – no Automatic Connection to WLAN • 0x00000001: WLAN Preferred – Automatic Connection to WLAN allowed • 0x00000002: : Automatic Connection to WLAN under conditions
AutomaticConnectionOptions	Input	Optional (if the Automatic Connection under condition is selected by the user)
HS20DefaultSubscription	Input	Name of the default HS2.0 WLAN operator subscription
ANDSFDefaultSubscription	Input	Optional – Name of the default ANDSF WLAN operator subscription – in case of dual SIM
userSetting	Input	Specifies the operator settings regarding which WLAN policies should apply • 0x00000000: No preference (i.e. operator settings applies) • 0x00000001: Cellular Preferred (i.e. ANDSF preferred) • 0x00000002: WLAN subscription preferred (i.e. HS2.0 Preferred) Any other value is interpreted as no preferences (equal to 0x00000000)

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.

7.13 Statistics APIs

7.13.1 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.

7.13.2 CMAPI_NetStatistic_GetAllConnectionRecords() - optional

The **CMAPI_NetStatistic_GetAllConnectionRecords ()** function is used to retrieve all connection records.

Prototype
dword **CMAPI_NetStatistic_GetAllConnRecord** (dword deviceID, ConnectionRecord* pConnRecord, dword* pConnRecordSize, dword* pConnRecordCount)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pConnRecord	Output	The pointer of connect record
pConnRecordSize	Input/Output	The size of the buffer on input or if insufficient contains the necessary size.
pConnRecordCount	Output	The total number of elements in the array of ConnRecord

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.
0X30000032	The buffer is not sufficient to hold the data, the pConnRecordSize 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.

7.13.3 CMAPI_NetStatistic_DeleteConnectionRecord() - optional

The **CMAPI_NetStatistic_DeleteConnectionRecord ()** function is used to delete a connection record.

Prototype
dword **CMAPI_NetStatistic_DeleteConnectionRecord** (dword deviceID, dword Index)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
Index	Input	The index of the record to be deleted

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 Information Status APIs

7.14.1 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.

7.14.2 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.

7.14.3 CMAPI_Information_GetSignalStrength()

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

Prototype
dword **CMAPI_Information_GetSignalStrength** (dword deviceID, RadioType Radio, UTF8* SSID, dword* pSignalStrengthRaw, dword* pSignalStrengthPercent, dword* pSignalQualityPercent)

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

deviceID	Input	The ID of the device concerned
Radio	Input	See RadioType definition
SSID	Input	(Optional) SSID of an Access Point Mandatory when the RadioType is WLAN
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.

7.14.4 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.

7.14.5 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.

7.14.6 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.

7.14.7 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.

7.14.8 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 • 0x00000000: Home • 0x00000001: 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.

7.14.9 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.

7.14.10 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.

7.14.11 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	The 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/Output	The size of the buffer in Bytes for the QoSContextlist or if 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.

7.14.12 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.

7.14.13 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.

7.14.14 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.

7.14.15 CMAPI_Information_GetBatteryStatus()

The **CMAPI_Information_GetBatteryStatus()** function is used to retrieve the current status of the battery of device if applicable.

Prototype
dword **CMAPI_Information_GetBatteryStatus** (dword deviceID, dword* pBatteryPresence, dword* pPowerSourceState, dword* pChargingState, dword* pBatteryLevel, dword* pRemainingTime)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pBatteryPresence	Output	The Indication if a battery present: • 0x00000000: No battery • 0x00000001: Battery present
pPowerSourceState	Output	The Indication of the power source state: • 0x00000001: External Source • 0x00000002: Battery

pChargingState	Output	The Indication of the charging state of the battery: • 0x00000000: Charging • 0x00000001: Not charging
pBatteryLevel	Output	Percentage of the battery available
pRemainingTime	Output	(Optional) Estimated Remaining time in minutes 0xFFFFFFFF not provided

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.

7.14.16 CMAPI_Information_SetBatteryThreshold()

The **CMAPI_Information_SetBatteryThreshold()** function is used to set thresholds for the battery status. If one of those thresholds is reached, the callback function **CMAPI_Callback_BatteryThresholdReached()** will be invoked.

The Upper level threshold is used to indicate that a certain percentage of the battery level is reached during charging.

The Lower level threshold is used to indicate that a certain percentage of the battery level is reached during discharging.

Prototype

dword **CMAPI_Information_SetBatteryThreshold** (dword deviceID, dword upperThreshold, dword lowerThreshold)

Parameters

Field Name	Mode	Description
deviceID	Input	The ID of the device concerned
upperThreshold	Input	The upper value of the battery level threshold in percentage
lowerThreshold	Input	The lower value of the battery level threshold in percentage

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
0X00000150	The threshold value(s) is/are 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.

7.14.17 CMAPI_Information_GetMobilityState() - Optional

The **CMAPI_Information_GetMobilityState()** function is used to retrieve the current mobility state of the device (the way the device get this information through sensors or other means is out of the scope of the document).

Prototype
dword **CMAPI_Information_GetMobilityState** (dword deviceID, dword* pMobilityState, dword* pMobilityAccuracy, dword* pSpeed, dword* pSpeedAccuracy)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pMobilityState	Output	The Indication of the mobility state of the device: • 0x00000001: Device is in Idle • 0x00000002: Device is in stationary mode • 0x00000004: Walking • 0x00000008: Running • 0x00000010: In moving vehicle (car, train, airplane...) • 0x00000020: Moving but pattern unknown
pMobilityAccuracy	Output	The estimated accuracy of the mobility state in percentage
pSpeed	Output	(Optional) The speed in meters per second
pSpeedAccuracy	Output	(Optional) The estimated accuracy of the speed in meters per second

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.

7.14.18 CMAPI_Information_GetMobilitytoLocation() - Optional

The **CMAPI_Information_GetMobilitytoLocation()** is used to evaluate if the device is moving compared to a specific location (for example, an Access Point). The calling thread returns immediately. The result is reported in callback **CMAPI_Callback_GetMobilitytoLocation_Complete().**

Prototype
dword **CMAPI_Information_GetMobilitytoLocation** (dword deviceID, GeoCircular* location, dword Timeout)

Parameters

Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
location	Input	The location to be evaluated
Timeout	Input	The maximum time for the evaluation of the movement compared to location specified (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.
0X00000151	The location 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 SMS Management APIs

7.15.1 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
0X4001XXXX	Operation cannot be done – Back off timer in place – time left is indicated by the 4 last digits (in seconds) XXXX is the time left in seconds - example 0X40010360 means 360 seconds are left

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_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.

7.15.3 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.

7.15.4 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* pIDList, dword* pIDListSize, dword* pIDListCount)

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
pIDList	Output	The list of dword values which reference SMS record identifiers.
pIDListSize	Input/Output	The size of the ID List buffer in bytes or if insufficient contains the necessary size.
pIDListCount	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 pIDList buffer is not sufficient, the pIDListSize 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.

7.15.5 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.

7.15.6 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.

7.15.7 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.

7.15.8 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.

7.15.9 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.

7.15.10 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.

7.15.11 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.

7.15.12 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.

7.15.13 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 unread SMS records

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.

7.15.14 CMAPI_SMS_Create()

The **CMAPI_SMS_Create()** function is used to create a draft SMS.

Prototype
dword **CMAPI_SMS_Create** (dword deviceID, dword systemID, SMSCreateRecord* pCreateRecord, 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
pCreateRecord	Input	The content of message to be saved.
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.
0X0000500D	The content of message to be saved 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.

7.16 USSD Management APIs

7.16.1 CMAPI_USSD_Request()

The **CMAPI_USSD_Request()** function is used to build up a USSD request to the network (see [3GPP TS 24.090]).

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.

7.16.2 CMAPI_USSD_Release()

The **CMAPI_USSD_Release()** function is used to release the USSD session (see [3GPP TS 24.090]), 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.

7.17 GNSS APIs

7.17.1 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.

7.17.2 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.

7.17.3 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.

7.17.4 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.

7.17.5 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.

7.17.6 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.

7.17.7 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.

7.17.8 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.

7.17.9 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.

7.17.10 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.

7.18 Data Push Service Management APIs

7.18.1 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.

7.18.2 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.

7.18.3 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.

7.19 Contact Management APIs

7.19.1 CMAPI_Contact_Create()

The **CMAPI_Contact_Create()** function is used to create a contact.

Prototype
dword **CMAPI_Contact_Create** (dword deviceID, dword systemID, ContactRecord* pContact)

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
pContact	Input/Output	The contact to be created (index is provided as 0x00000000 as input and value of the index will be provided as output)

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.
0X00005501	The contact record is invalid
0X00005502	Memory capacity exceeded.
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.

7.19.2 CMAPI_Contact_Get()

The **CMAPI_Contact_Get()** function is used to retrieve the details of a contact.

Prototype
dword **CMAPI_Contact_Get** (dword deviceID, dword systemID, dword index, dword contactLocation, ContactRecord* 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
index	Input	The index of the contact
contactLocation	Input	To indicate where the contact record is • 0x00000000: from SIM/R-UIM/NAA on UICC • 0x00000001: from local device • 0x00000002: from the terminal device, like PC
pRecord	Output	The contact 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.
0X00000003	Buffer size 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.
0X00005007	The ifrom value is invalid
0X00005503	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.

7.19.3 CMAPI_Contact_Delete()

The **CMAPI_Contact_Delete()** function is used to delete a contact.

Prototype
dword **CMAPI_Contact_Delete** (dword deviceID, dword systemID, dword index, dword contactLocation)

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
index	Input	The index of the contact
contactLocation	Input	To indicate where the contact is stored • 0x00000000: SIM/R-UIM/NAA on UICC • 0x00000001: local device • 0x00000002: the terminal device, like PC

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.
0X00005503	The index is invalid
0X00005504	The contact location 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.

7.19.4 CMAPI_Contact_GetContactList()

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

Prototype
dword CMAPI_Contact_GetContactList (dword deviceID, dword systemID, dword contactLocation, ContactRecords* pContactList, dword* pSize)

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
contactLocation	Input	To indicate where the contact is stored • 0x00000000: SIM/R-UIM/NAA on UICC • 0x00000001: local device • 0x00000002: the terminal device, like PC
pContactList	Output	The pointer of contact list
pSize	Input/Output	The size of the contact list or if insufficient contains the necessary size

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000003	Buffer size 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.
0X00005504	The contact location 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.

7.19.5 CMAPI_Contact_Update()

The **CMAPI_Contact_Update()** is used to update an existing contact.

Prototype
dword **CMAPI_Contact_Update** (dword deviceID, dword systemID, ContactRecord* 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 contact 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.
0X00005501	The contact 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.

7.19.6 CMAPI_Contact_Search()

The **CMAPI_Contact_Search()** function is used to search for a specific contact name in the list of contacts.

Prototype
dword **CMAPI_Contact_Search** (dword deviceID, dword systemID, UTF8* tomatch, dword* pMatchfound, ContactRecords* pRecordList, dword* pRecordListSize)

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
tomatch	Input	The characters to be searched in all the fields of the records
pMatchfound	Output	The status of the contact name searched • 0x00000000: No match found • 0x00000001: Match(es) found
pRecordList	Output	The contact record list having fields matching the request (several records can match this request)
pRecordListSize	Input/Output	The size of the record list buffer or if insufficient contains the necessary size

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000003	Buffer size 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.
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.

7.20 P2P Direct Management APIs

7.20.1 CMAPI_P2P_GetP2PInfo()

The **CMAPI_P2P_GetP2PInfo()** function is used to detect which P2P direct connection technology(ies) is/are supported if any.

Prototype
dword **CMAPI_P2P_GetP2PInfo**(dword deviceID, P2PInfoType* pP2PInfo)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
pP2PInfo	Output	Pointer to get the P2P direct connection technology type in bitmap. In the case of a device with multiple P2P direct connection technologies supported, there MAY be multiple settings returned. The bitmap definition follows the definition of P2PInfoType: • 0x00000000: None • 0x00000001: Wi-Fi Direct (could there be several versions) • 0x00000002: reserved for future use • 0x00000004: reserved for future use • 0x00000008: reserved for future use • 0x00000010: reserved for future use • 0x00010000: LTE Direct • 0x00020000: reserved for future use

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.

7.20.2 CMAPI_P2P_EnableDirectDiscovery()

The **CMAPI_P2P_EnableDirectDiscovery()** function is used to activate the P2P Direct Discovery Feature in a P2P Direct enabled device.

Prototype
dword **CMAPI_P2P_EnableDirectDiscovery** (dword deviceID, P2PinfoType P2PTechno)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
P2PTechno	Input	The P2P direct connection technology to enable Please see P2PInfoType 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.
0X00000130	The device is not in a power state which allows this operation.
0X00008001	The P2PTechnology is not supported
0X00008002	The P2P Technology 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.

7.20.3 CMAPI_P2P_DisableDirectDiscovery()

The **CMAPI_P2P_DisableDirectDiscovery()** function is used to deactivate the P2P Direct Discovery feature in a P2P Direct enabled device.

Prototype
dword **CMAPI_P2P_DisableDirectDiscovery** (dword deviceID, P2PinfoType P2PTechno)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
P2PTechno	Input	The P2P direct connection technology to disable Please see P2PInfoType 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.
0X00000130	The device is not in a power state which allows this operation.
0X00008001	The P2PTechnology is not supported
0X00008002	The P2P Technology 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.

7.20.4 CMAPI_P2P_EnableDirectConnection()

The **CMAPI_P2P_EnableDirectConnection()** function is used to activate the P2P Direct Connection feature in a P2P Direct enabled device.

Prototype
dword **CMAPI_P2P_EnableDirectConnection** (dword deviceID, P2PinfoType P2PTechno)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
P2PTechno	Input	The P2P direct connection technology to enable Please see P2PInfoType 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.
0X00000130	The device is not in a power state which allows this operation.
0X00008001	The P2PTechnology is not supported
0X00008002	The P2P Technology 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.

7.20.5 CMAPI_P2P_DisableDirectConnection()

The **CMAPI_P2P_DisableDirectConnection()** function is used to deactivate the P2P Direct Connection feature in a P2P Direct enabled device.

Prototype
dword **CMAPI_P2P_DisableDirectConnection** (dword deviceID, P2PinfoType P2PTechno)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
P2PTechno	Input	The P2P direct connection technology to enable Please see P2PInfoType 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.
0X00000130	The device is not in a power state which allows this operation.
0X00008001	The P2PTechnology is not supported
0X00008002	The P2P Technology 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.

7.20.6 CMAPI_P2P_DiscoveryResolve()

The **CMAPI_P2P_DiscoveryResolve()** function is used to resolve a ServiceRecord for metadata and/or connection info.

Prototype
dword **CMAPI_P2P_DiscoveryResolve** (dword deviceID, ServiceRecord* pService, P2PInfoType* pP2PInfo)

Parameters

Field Name	Mode	Description
deviceID	Input	The ID of the device concerned.
pService	Input	The ServiceRecord to be resolved.
pP2PInfo	Output	Pointer to get the P2P direct connection technology type in bitmap. In the case of a device with multiple P2P direct connection technologies supported, there MAY be multiple settings returned. The bitmap definition follows the definition of P2PInfoType: • 0x00000000: None • 0x00000001: Wi-Fi Direct (could there be several versions) • 0x00000002: reserved for future use • 0x00000004: reserved for future use • 0x00000008: reserved for future use • 0x00000010: reserved for future use • 0x00010000: LTE Direct • 0x00020000: reserved for future use

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.
0X00008003	The Service 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.

7.20.7 CMAPI_P2P_DiscoveryMonitor()

The **CMAPI_P2P_DiscoveryMonitor()** function is used to request discovery of Remote Device(s) and the services offered.

Prototype
dword **CMAPI_P2P_DiscoveryMonitor** (dword deviceID, DeviceRecords* pRemoteDevices, ServiceRecords* pServices)

Parameters

Field Name	Mode	Description
deviceID	Input	The ID of the device concerned.
pRemoteDevices	Input	Optional; List of Remote Devices to be discovered
pServices	Input	List of Service Identifiers to be discovered

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.
0X00008004	The list of Remote Devices is invalid.
0X00008005	The list of Service Identifiers 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.

7.20.8 CMAPI_P2P_DiscoveryAnnounce()

The **CMAPI_P2P_DiscoveryAnnounce()** function is used by applications in order to announce its P2P Direct services supported or by the Local Device to announce its presence to Remote Device(s).

Prototype
dword **CMAPI_P2P_DiscoveryAnnounce** (dword deviceID, DeviceRecords* pRemoteDevices, ServiceRecords* pServices)

Parameters

Field Name	Mode	Description
deviceID	Input	The ID of the device concerned.
pRemoteDevices	Input	Optional; List of Remote Devices addressed by this announcement
pServices	Input	List of Service Identifiers announced by Local Device

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.
0X00008004	The list of Remote Devices is invalid.
0X00008005	The list of Service Identifiers 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.

7.20.9 CMAPI_P2P_EstablishConnection()

The **CMAPI_P2P_EstablishConnection()** function is used to request the Local Device to establish a connection (P2P Direct connection or connection via network, subject to Service Provider policy) to a Remote Device.

Prototype
dword **CMAPI_P2P_EstablishConnection** (dword deviceID, dword remoteDeviceID, ServiceRecord service, dword* pConnectionID)

Parameters

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

remoteDeviceID	Input	Optional; The ID of the Remote Device concerned
service	Input	Service Identifier
pConnectionID	Output	The ID of the connection 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.
0X00008004	The list of Remote Devices is invalid.
0X00008005	The list of Service Identifiers 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.

7.20.10 CMAPI_P2P_RejectConnection()

The **CMAPI_P2P_RejectConnection()** function is used to reject an incoming connection request (P2P Direct connection or connection via network, subject to Service Provider policy).

Prototype

dword **CMAPI_P2P_RejectConnection** (dword deviceID, dword connectionID)

Parameters

Field Name	Mode	Description
deviceID	Input	The ID of the device concerned.
connectionID	Input	The ID of the connection 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.
0X00008006	The ID of the Connection 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.

7.20.11 CMAPI_P2P_AcceptConnection()

The **CMAPI_P2P_AcceptConnection()** function is used to accept an incoming connection request (P2P Direct connection or connection via network, subject to Service Provider policy) from a Remote Device.

Prototype
dword **CMAPI_P2P_AcceptConnection** (dword deviceID, dword connectionID)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
connectionID	Input	The ID of the connection 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.
0X00008006	The ID of the Connection 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.

7.20.12 CMAPI_P2P_CloseConnection()

The **CMAPI_P2P_CloseConnection()** function is used to request the Local Device to close an existing connection (P2P Direct connection or connection via network, subject to Service Provider policy) to a Remote Device.

Prototype
dword **CMAPI_P2P_CloseConnection** (dword deviceID, dword connectionID)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
connectionID	Input	The ID of the connection 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.
0X00008006	The ID of the Connection 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.

7.20.13 CMAPI_P2P_GetConnectionStatus()

The CMAPI_P2P_GetConnectionStatus() function is used to retrieve the status of the P2P Direct connection.

Prototype
dword **CMAPI_P2P_GetConnectionStatus** (dword deviceID, dword connectionID, dword* pConnectionStatus)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
connectionID	Input	The ID of the connection concerned
pConnectionStatus	Output	Status of the connection • Connected • Connecting

		• Disconnected • Disconnecting

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.
0X00008006	The ID of the Connection 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.

7.20.14 CMAPI_P2P_EnableRelay()

The **CMAPI_P2P_EnableRelay()** function is used to request the Local Device to act as a relay to share its data connection with Remote Device members of the group (i.e. enable concurrent operations).

Prototype
dword **CMAPI_P2P_EnableRelay** (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.

7.20.15 CMAPI_P2P_DisableRelay()

The **CMAPI_P2P_DisableRelay()** function is used able to request the Local Device to stop acting as a relay to share its data connection with Remote Device members of the group (i.e. disable concurrent operations).

Prototype
dword **CMAPI_P2P_DisableRelay** (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.

7.20.16 CMAPI_P2P_CreateGroup()

The **CMAPI_P2P_CreateGroup()** function is used to create a new P2P Direct group with one or several Remote Device (s) (The group could be a simple instance group – one time or a persistent one).

Prototype
dword **CMAPI_P2P_CreateGroup** (dword deviceID, DeviceRecords* pDevices, dword* pGroupID)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
pDevices	Input	The list of device IDs belonging to this group
pGroupID	Input/Output	The ID of the group created

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.
0X00008007	The list of Device ID is invalid
0X00008008	The ID of the group 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.

7.20.17 CMAPI_P2P_RemoveGroup()

The **CMAPI_P2P_RemoveGroup()** function is used to remove a P2P group, previously created by the Local Device.

Prototype
dword **CMAPI_P2P_RemoveGroup** (dword deviceID, dword groupID)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
groupID	Input	The ID of the group to be removed

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.
0X00008008	The ID of the group 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.

7.20.18 CMAPI_P2P_EnableMembershipInSeveralGroups()

The **CMAPI_P2P_EnableMembershipInSeveralGroups()** function is used to enable a Local Device to be a member of several groups simultaneously.

Prototype
dword **CMAPI_P2P_EnableMembershipInSeveralGroups** (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.

7.20.19 CMAPI_P2P_DisableMembershipInSeveralGroups()

The **CMAPI_P2P_DisableMembershipInSeveralGroups()** function is used to disable a Local Device to be a member of several groups simultaneously, providing that the Local Device is not in charge of any of these groups.

Prototype
dword **CMAPI_P2P_DisableMembershipInSeveralGroups** (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.

7.20.20 CMAPI_P2P_RemoveDeviceFromGroup()

The **CMAPI_P2P_RemoveDeviceFromGroup()** function is used by the Local Device to remove a Remote Device from an existing group the Local Device owns.

Prototype
dword **CMAPI_P2P_RemoveDeviceFromGroup** (dword deviceID, dword groupID, dword remoteDeviceID)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
groupID	Input	The ID of the group the device should be removed from
remoteDeviceID	Input	The ID of the remote device to be removed from the group.

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.
0X00008008	The ID of the group is invalid
0X00008009	The ID of the Remote Device 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.

7.20.21 CMAPI_P2P_AcceptInvitationToGroup()

The **CMAPI_P2P_AcceptInvitationToGroup()** function is used to positively accept an group join invitation on the receiver side.

Prototype
dword **CMAPI_P2P_AcceptInvitationToGroup**(dword deviceID, dword groupID, dword remoteDeviceID, dword invitationID)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
groupID	Input	The ID of the group the Remote Device accepts to join.
remoteDeviceID	Input	The ID of the remote device accepting the invitation to this group.
invitationID	Input	The ID of this invitation

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.
0X00008008	The ID of the group is invalid
0X00008009	The ID of the Remote Device is invalid
0X0000800A	The Invitation ID 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.

7.20.22 CMAPI_P2P_JoinGroup()

The **CMAPI_P2P_JoinGroup()** function is used to invite a Remote Device to join an existing group.

Prototype
dword **CMAPI_P2P_JoinGroup** (dword deviceID, dword groupID, dword remoteDeviceID, dword invitationID)

Parameters

Field Name	Mode	Description
deviceID	Input	The ID of the device concerned.
groupID	Input	The ID of the group the Remote Device is invited to join
remoteDeviceID	Input	The ID of the remote device being invited to this group.
invitationID	Input	The ID of this invitation

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.
0X00008008	The ID of the group is invalid
0X00008009	The ID of the Remote Device is invalid
0X0000800A	The Invitation ID 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.

7.20.23 CMAPI_P2P_RejectInvitationToGroup()

The **CMAPI_P2P_RejectInvitationToGroup()** function is used to reject an invitation to join an existing group.

Prototype

dword **CMAPI_P2P_RejectInvitationToGroup** (dword deviceID, dword groupID, dword remoteDeviceID, dword invitationID)

Parameters

Field Name	Mode	Description
deviceID	Input	The ID of the device concerned.
groupID	Input	The ID of the group the Local Device is rejecting to join
remoteDeviceID	Input	The ID of the device rejecting the invitation to join this group.
invitationID	Input	The ID of the invitation 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.
0X00008008	The ID of the group is invalid
0X00008009	The ID of the Remote Device is invalid
0X0000800A	The Invitation ID 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.

7.20.24 CMAPI_P2P_RejectJoiningGroup()

The **CMAPI_P2P_RejectJoiningGroup()** function is used to reject a Remote Device from joining to an existing group.

Prototype
dword **CMAPI_P2P_RejectJoiningGroup** (dword deviceID, dword groupID, dword remoteDeviceID, dword requestID)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
groupID	Input	The ID of the group concerned
remoteDeviceID	Input	The ID of the remote device being rejected to join this group.
requestID	Input	The ID of the request for joining the group.

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.

0X00008008	The ID of the group is invalid
0X00008009	The ID of the Remote Device is invalid
0X0000800A	The Invitation ID 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.

7.20.25 CMAPI_P2P_RequestToJoinGroup()

The **CMAPI_P2P_RequestToJoinGroup()** function is used to send a request for joining an existing group to the group owner.

Prototype
dword **CMAPI_P2P_RequestToJoinGroup** (dword deviceID, dword groupID, dword remoteDeviceID, dword requestID)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned.
groupID	Input	The ID of the group the device wants to join
remoteDeviceID	Input	The ID of the device being group owner.
requestID	Input	The ID of this 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.
0X00008008	The ID of the group is invalid
0X00008009	The ID of the Remote Device 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.

7.20.26 CMAPI_P2P_RestrictFromGroup()

The **CMAPI_P2P_RestrictFromGroup()** function is used to instruct the Local Device to be restricted from an existing group owned by a Remote Device.

Prototype
dword **CMAPI_P2P_RestrictFromGroup**(dword deviceID, dword groupID, dword remoteDeviceID)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned (Local Device).
groupID	Input	The ID of the group the Local Device to be restricted from.
remoteDeviceID	Input	The ID of the remote device owning this group.

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.
0X00008008	The ID of the group is invalid
0X00008009	The ID of the Remote Device 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.

7.20.27 CMAPI_P2P_GetGroupInfo()

The **CMAPI_P2P_GetGroupInfo()** function is used to retrieve from the Local Device which P2P Direct enabled device(s) are in an existing group to which the Local Device is a member of.

Prototype
dword **CMAPI_P2P_GetGroupInfo** (dword deviceID, dword groupID, DeviceRecords* pDeviceList)

Parameters

Field Name	Mode	Description
deviceID	Input	The ID of the device concerned.
groupID	Input	The ID of the group concerned.
pDeviceList	Output	The list of member devices of the group 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.
0X00008008	The ID of the group 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.

7.20.28 CMAPI_P2P_AllowSimultaneousConnection()

The **CMAPI_P2P_AllowSimultaneousConnection()** function is used to allow the device to have a P2P connection simultaneously to a normal data connection using the same radio technology.

Prototype

dword **CMAPI_P2P_AllowSimultaneousConnection** (dword deviceID, P2PinfoType P2PTechno)

Parameters

Field Name	Mode	Description
deviceID	Input	The ID of the device concerned.
P2PTechno	Input	The P2P direct connection technology to enable Please see P2PInfoType 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.

0X00000130	The device is not in a power state which allows this operation.
0X00008001	The P2PTechnology is not supported
0X00008002	The P2P Technology 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.

7.20.29 CMAPI_P2P_DisallowSimultaneousConnection()

The **CMAPI_P2P_DisallowSimultaneousConnection()** function is used to disallow the device to have a P2P connection simultaneously to a normal data connection using the same radio technology.

Prototype
dword **CMAPI_P2P_DisallowSimultaneousConnection** (dword deviceID, P2PinfoType P2PTechno)

Parameters

Field Name	Mode	Description
deviceID	Input	The ID of the device concerned.
P2PTechno	Input	The P2P direct connection technology to disable Please see P2PInfoType 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.
0X00000130	The device is not in a power state which allows this operation.
0X00008001	The P2PTechnology is not supported
0X00008002	The P2P Technology 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.

7.21 Wireless Router APIs

This section describes APIs related to controlling a wireless router and managing its Connected Devices. A physical router device may contain multiple router configurations to manage different networks. Use of the WWAN broadband data connection by Connected Devices is controlled by policies and restrictions. APIs are included for router configuration, setting policies and restrictions for Connected Devices, and administrator security.

7.21.1 CMAPI_Router_GetConfigurations()

The **CMAPI_Router_GetConfigurations()** function is used to read the configuration values of a router (ssid, users, security, etc) of all defined routers of a physical router device. See RouterConfigType for a description of configuration parameters.

Prototype
dword **CMAPI_Router_GetConfigurations** (dword deviceID, RouterConfigType* pRouterConfigList, dword* pRouterConfigListSize, word* pRouterConfigListElements)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pRouterConfigList	Output	The list of router configurations. See RouterConfigType for details of the returned data.
pRouterConfigListSize	Input/Output	The number of bytes in the pRouterConfigList buffer on input or if insufficient contains the necessary size.
pRouterConfigListElements	Output	The number of elements in the router list.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device.
0X00000104	The device does not contain hardware which supports this operation.
0X00000130	The device is not in a power state which allows this operation.
0X30009001	The pRouterConfigList 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.
0XF0000003	The authentication has been denied. Please seek proper credentials for your access level.

7.21.2 CMAPI_Router_SetConfiguration()

The **CMAPI_Router_SetConfiguration()** function is used to write the configuration values of a router (ssid, users, security, etc). If a router ID does not currently exist, the router configuration is created. See RouterConfigType for a description of configuration parameters.

Prototype
dword **CMAPI_Router_SetConfiguration** (dword deviceID, dword routerID, RouterConfigType* pRouterConfig)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
routerID	Input	The ID of the router concerned
pRouterConfig	Input	Configuration values of the router. See RouterConfigType definition

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device.
0X00000104	The device does not contain hardware which supports this operation.
0X00000112	The routerID references a non-existing router.
0X00000130	The device is not in a power state which allows this operation.
0X00009001	The routerConfig value(s) are incorrect
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.
0XF0000003	The authentication has been denied. Please seek proper credentials for your access level.

7.21.3 CMAPI_Router_DeleteConfiguration()

The **CMAPI_Router_DeleteConfiguration()** function is used to delete a router and its configuration. See RouterConfigType for a description of configuration parameters.

Prototype
dword **CMAPI_Router_DeleteConfiguration** (dword deviceID, dword routerID)

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

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device.
0X00000104	The device does not contain hardware which supports this operation.
0X00000112	The routerID references a non-existing router
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.
0XF0000003	The authentication has been denied. Please seek proper credentials for your access level.

7.21.4 CMAPI_Router_GetConnectedDevices()

The **CMAPI_Router_GetConnectedDevices()** function is used to retrieve a list of Connected Devices connected to a router. The list contains records with Connected Device information.

Prototype
dword **CMAPI_Router_GetConnectedDevices** (dword deviceID, dword routerID, ConnectedDevType* pConDevList, dword* pConDevListSize, word* pConDevListElements)

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

pConDevList	Output	The list of connected devices. See ConnectedDevType for details of the returned data.
pConDevListSize	Input/Output	The number of bytes in the pConDevList buffer on input or if insufficient contains the necessary size.
pConDevListElements	Output	The number of elements in the Connected Device list.

Return Values

Value	Description
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device.
0X00000104	The device does not contain hardware which supports this operation.
0X00000112	The routerID references a non-existing router
0X00000130	The device is not in a power state which allows this operation.
0X30009002	The pConDevList 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.
0XF0000003	The authentication has been denied. Please seek proper credentials for your access level.

7.21.5 CMAPI_Router_GetPolicies()

The **CMAPI_Router_GetPolicies()** function is used to retrieve a list of policies within a router. The list contains records with policy information.

Prototype

dword **CMAPI_Router_GetPolicies** (dword deviceID, dword routerID, PolicyType* pPolicyList, dword* pPolicyListSize, word* pPolicyListElements)

Parameters

Field Name	Mode	Description
deviceID	Input	The ID of the device concerned
routerID	Input	The ID of the router concerned
pPolicyList	Output	The list of policies. The list contains the default policy and optional user policies. See PolicyType for details of the returned data.
pPolicyListSize	Input/Output	The number of bytes in the pPolicyList buffer on input or if insufficient contains the necessary size.

pPolicyListElements	Output	The number of elements in the policy list.

Return Values

Value	Description
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device.
0X00000104	The device does not contain hardware which supports this operation.
0X00000112	The routerID references a non-existing router
0X00000130	The device is not in a power state which allows this operation.
0X30009003	The pPolicyList 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.
0XF0000003	The authentication has been denied. Please seek proper credentials for your access level.

7.21.6 CMAPI_Router_SetPolicy()

The **CMAPI_Router_SetPolicy()** function is used to add or update a policy to a router's policies. See PolicyType for a description of policy parameters.

Prototype

dword **CMAPI_Router_SetPolicy** (dword deviceID, dword routerID, PolicyType* pPolicy)

Parameters

Field Name	Mode	Description
deviceID	Input	The ID of the device concerned
routerID	Input	The ID of the router concerned
pPolicy	Input	A router policy. See PolicyType definition

Return Values

Value	Description
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device.

0X00000104	The device does not contain hardware which supports this operation.
0X00000112	The routerID references a non-existing router
0X00000130	The device is not in a power state which allows this operation.
0X00009002	The policy value(s) are incorrect
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.
0XF0000003	The authentication has been denied. Please seek proper credentials for your access level.

7.21.7 CMAPI_Router_DeletePolicy()

The **CMAPI_Router_DeletePolicy()** function is used to delete a Connected Device policy from a router's policies. See PolicyType for a description of policy parameters.

Prototype
dword **CMAPI_Router_DeletePolicy** (dword deviceID, dword routerID, UTF8* MACAddress)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
routerID	Input	The ID of the router concerned
MACAddress	Input	The MAC address of the Connected Device

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device.
0X00000104	The device does not contain hardware which supports this operation.
0X00000112	The routerID references a non-existing router
0X00000130	The device is not in a power state which allows this operation.
0X00000140	The MACAddress references a non-existing Connected Device
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.
0XF0000003	The authentication has been denied. Please seek proper credentials for your access level.

7.21.8 CMAPI_Router_GetRestrictions()

The **CMAPI_Router_GetRestrictions()** function is used to retrieve a list of Connected Device restrictions within a router. The list contains records with blacklisted and blocked information.

Prototype
dword **CMAPI_Router_GetRestrictions** (dword deviceID, dword routerID, RestrictType* pRestrictList, dword* pRestrictListSize, word* pRestrictListElements)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
routerID	Input	The ID of the router concerned
pRestrictList	Output	The list of restrictions. See RestrictType for details of the returned data.
pRestrictListSize	Input/Output	The number of bytes in the pRestrictList buffer on input or if insufficient contains the necessary size.
pRestrictListElements	Output	The number of elements in the restriction list.

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device.
0X00000104	The device does not contain hardware which supports this operation.
0X00000112	The routerID references a non-existing router
0X00000130	The device is not in a power state which allows this operation.
0X30009004	The pRestrictList 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.
0XF0000003	The authentication has been denied. Please seek proper credentials for your access level.

7.21.9 CMAPI_Router_SetRestriction()

The **CMAPI_Router_SetRestriction()** function is used to add or update a Connected Device restriction to a router. See RestrictType for a description of restriction parameters.

Prototype

dword **CMAPI_Router_SetRestriction** (dword deviceID, dword routerID, RestrictType* pRestrict)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
routerID	Input	The ID of the router concerned
pRestrict	Input	A router restriction. See RestrictType definition

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device.
0X00000104	The device does not contain hardware which supports this operation.
0X00000112	The routerID references a non-existing router
0X00000130	The device is not in a power state which allows this operation.
0X00009003	The restrict value(s) are incorrect
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.
0XF0000003	The authentication has been denied. Please seek proper credentials for your access level.

7.21.10 CMAPI_Router_DeleteRestriction()

The **CMAPI_Router_DeleteRestriction()** function is used to remove a Connected Device restriction. See RestrictType for a description of restriction parameters.

Prototype
dword **CMAPI_Router_DeleteRestriction** (dword deviceID, dword routerID, UTF8* MACAddress)

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

MACAddress	Input	The MAC address of the Connected Device

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device.
0X00000104	The device does not contain hardware which supports this operation.
0X00000112	The routerID references a non-existing router
0X00000130	The device is not in a power state which allows this operation.
0X00000140	The MACAddress references a non-existing Connected Device
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.
0XF0000003	The authentication has been denied. Please seek proper credentials for your access level.

7.21.11 CMAPI_Router_SetAdminPassword()

The **CMAPI_Router_SetAdminPassword()** function is used to update a router administrator password.

Prototype
dword **CMAPI_Router_SetAdminPassword** (dword deviceID, dword routerID, UTF8* pAdminPass)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
routerID	Input	The ID of the router concerned
pAdminPass	Input	The administrator password

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

0X00000112	The routerID references a non-existing router
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.
0XF0000003	The authentication has been denied. Please seek proper credentials for your access level.

7.21.12 CMAPI_Router_VerifyAdminPassword()

The **CMAPI_Router_VerifyAdminPassword()** function is used to verify a router administrator password and to report the number of failed access attempts.

Prototype
dword **CMAPI_Router_VerifyAdminPassword** (dword deviceID, dword routerID, UTF8* pAdminPass, dword* pFailedAttempts)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
routerID	Input	The ID of the router concerned
pAdminPass	Input	The administrator password
pFailedAttempts	Output	The number of failed password verification attempts

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device.
0X00000104	The device does not contain hardware which supports this operation.
0X00000112	The routerID references a non-existing router
0X00000130	The device is not in a power state which allows this operation.
0X00009004	The administrator password is incorrect
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.
0XF0000003	The authentication has been denied. Please seek proper credentials for your access level.

7.21.13 CMAPI_Router_ResetToDefaults()

The **CMAPI_Router_ResetToDefaults()** function returns a router to factory default settings.

Prototype
dword **CMAPI_Router_ResetToDefaults** (dword deviceID, dword routerID)

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

Return Values	
Value	**Description**
0X00000000	The function succeeded.
0X00000001	A fatal error has occurred.
0X00000101	The deviceID references a non-existing device.
0X00000104	The device does not contain hardware which supports this operation.
0X00000112	The routerID references a non-existing router
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.
0XF0000003	The authentication has been denied. Please seek proper credentials for your access level.

7.22 IP Multimedia Services

7.22.1 CMAPI_IMS_GetISIMinfo()

The **CMAPI_IMS_GetISIMinfo()** function is used to retrieve if there is an ISIM in the UICC for a specific radio system (either 3GPP or 3GPP2) and to provide the IMPU, IMPI & Home Domain Name (relevant for IMS context) related.

Prototype
dword **CMAPI_IMS_GetISIMInfo** (dword deviceID, dword systemID, dword* pISIMpresent, IMPUs* pIMPU, dword* pIMPUSize, UTF8* pIMPI, UTF8* pHomeDomainName)

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
pISIMpresent	Output	The Indication if there is ISIM in the UICC for this radio system: • 0x00000000: No ISIM in the UICC • 0x00000001: ISIM in the UICC
pIMPU	Output	The IMS Public User Identities of the ISIM for this radio system if present. If ISIM is not present, a temporary IMPU based on IMSI, MCC and MNC is created in accordance with rules defined in [3GPP TS 23.003])
pIMPUSize	Input/Output	The size of the pIMPU buffer in bytes.
pIMPI	Output	The IMS Private User Identity of the ISIM for this radio system if present. If ISIM is not present, a temporary IMPI based on IMSI, MCC and MNC is created in accordance with rules defined in [3GPP TS 23.003])
pHomeDomainName	Output	The Home Domain Name of the ISIM for this radio system if present. If ISIM is not present, a temporary Home Domain Name based on IMSI, MCC and MNC is created in accordance with rules defined in [3GPP TS 23.003])

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.
0X00000555	The terminal response is invalid
0X30001001	The pIMPU buffer is not large enough. pIMPUSize 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.

7.22.2 CMAPI_IMS_GetIARInfo()

The **CMAPI_IMS_GetIARIinfo()** function is used to retrieve the IARI information from the ISIM for a dedicated radio system (either 3GPP or 3GPP2) on the UICC.

Prototype
dword **CMAPI_IMS_GetIARIInfo** (dword deviceID, dword systemID, UTF8* pIARI, dword* pIARISize)

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
pIARI	Output	The IMS Application Reference IDs coded as URN in accordance with [3GPP TS 24.229] for this radio system. A list of the URNs containing IARI values registered by 3GPP can be found at http://www.3gpp.com/Uniform-Resource-Name-URN-list.html
pIARISize	Input/Output	The size of the pIARI 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.

0X00000107	System not supported by the device
0X00000130	The device is not in a power state which allows this operation.
0X00000555	The terminal response is invalid
0X30001002	The pIARI buffer is not large enough. pIARISize 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.

7.23 M2M/IoT APIs

7.23.1 CMAPI_IoT_IMSI_Attach()

The **CMAPI_IoT_IMSI_Attach()** function is used to request an IMSI attach or detach.

Prototype
dword **CMAPI_IoT_IMSI_Attach** (dword deviceID, dword IMSIRegistration)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
IMSIRegistration	Input	To proceed to IMSI attach or detach: • 0x00000000: detach • 0x00000001: attach

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.
0X4001XXXX	Operation cannot be done – Back off timer in place – time left is indicated by the 4 last digits (in seconds)
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.

7.23.2 CMAPI_IoT_GPRS_Register()

The **CMAPI_IoT_GPRS_Register()** function is used to request an GPRS attach or detach.

Prototype
dword **CMAPI_IoT_GPRS_Register** (dword deviceID, dword GPRSRegistration)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
GPRSRegistration	Input	To attach or detach to GPRS: • 0x00000000: detach • 0x00000001: attach

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.
0X4001XXXX	Operation cannot be done – Back off timer in place – time left is indicated by the 4 last digits (in seconds)
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.

7.23.3 CMAPI_IoT_Set_PDPContext()

The **CMAPI_IoT_Set_PDPContext()** function is used to define a PDP context.

Prototype
dword **CMAPI_IoT_Set_PDPContext** (dword deviceID, dword PDPContextID, dword PDPType, UTF8* APN)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
PDPContextID	Input	The numerical identification of the PDP Context. The parameter is local to the modem interface and is used in other PDP context-related functions.
PDPType	Input	The type of PDP (Packet Data Protocol): • 0x00000001: IP • 0x00000002: PPP - PS data over GPRS or UMTS (PS connection with PDP type PPP)

		• 0x00000004: IPv6 • 0x00000008:X.25
APN	Input	Primary APN used for this connection

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.
0X00000160	The PDP context ID is invalid
0X00000161	The PDP Type is invalid
0X4001XXXX	Operation cannot be done – Back off timer in place – time left is indicated by the 4 last digits (in seconds).
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.

7.23.4 CMAPI_IoT_GetPDPContextList()

The **CMAPI_IoT_GetPDPContextList ()** function is used to get the list of currently defined PDP Contexts.

Prototype
dword **CMAPI_IoT_GetPDPContextList** (dword deviceID, PDPContext* pPDPContextList, dword* pPDPContextListSize, dword* pPDPContextListCount)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pPDPContextList	Output	The List of PDP contexts
pPDPContextListSize	Input/Output	The size of the buffer on input or if insufficient contains the necessary size.
pPDPContextListCount	Output	The total number of elements in the array of pPDPContextList

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.
0X30000033	The buffer is not sufficient to hold the data, the pPDPContextListSize 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.

7.23.5 CMAPI_IoT_GetPDPContextIPAddress()

The **CMAPI_IoT_GetPDPContextIPaddress ()** function is used to retrieve the IP address of the PDP context concerned.

Prototype
dword **CMAPI_IoT_GetPDPContextIPAddress** (dword deviceID, dword contextID, UTF8* pPDPContextIPaddress)

Parameters

Field Name	Mode	Description
deviceID	Input	The ID of the device concerned
contextID	Input	The PDP context concerned
pPDPContextIPaddress	Output	The IP address of the PDP context concerned

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.
0X00000160	The PDP context ID 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.

7.23.6 CMAPI_IoT_Activate_PDPContext()

The **CMAPI_IoT_Activate_PDPContext()** function is used to activate or deactivate a PDP context.

Prototype
dword **CMAPI_IoT_Activate_PDPContext** (dword deviceID, dword state, dword contextID)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
state	Input	To activate or deactivate a context: • 0x00000000: deactivate • 0x00000001: activate
contextID	Input	The PDP context 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.
0X00000160	The PDP context ID is invalid
0X4001XXXX	Operation cannot be done – Back off timer in place – time left is indicated by the 4 last digits (in seconds).
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.

7.23.7 CMAPI_IoT_SetNFM()

The **CMAPI_IoT_SetNFM()** function is used to set up (enable or disable) the Network Friendly Mode of the modem if supported.

Prototype
dword **CMAPI_IoT_SetNFM** (dword deviceID, dword setNFM)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
setNFM	Input	To enable or disable the Network Friendly Mode of the modem: • 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.
0X00000121	The device does not offer this capability
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.

7.23.8 CMAPI_IoT_GetNFM()

The **CMAPI_IoT_GetNFM()** function is used to return the current state of the Network Friendly Mode of the modem (enabled or disabled).

Prototype
dword **CMAPI_IoT_GetNFM** (dword deviceID, dword* pStateNFM)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pStateNFM	Output	To indicate if the Network Friendly Mode 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.
0X00000121	The device does not offer this capability
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.

7.23.9 CMAPI_IoT_SetBack-OffBaseInterval()

The **CMAPI_IoT_SetBack-OffBaseInterval()** function is used to configure the Back-off Base Intervals of the modem (time between re-attempts of whatever action previously failed).

Prototype
dword **CMAPI_IoT_SetBack-OffBaseInterval** (dword deviceID, UTF8* Back-offBaseIntervals)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
Back-offBaseIntervals	Input	The Back-off Base Intervals to be set up. Each interval is interpreted in amount of seconds (of maximum four digits) and should be separated by the character "," . Example: 60, 120, 240, 480, 960, 1920, 3840

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.
0X00000164	The back off time interval 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.

7.23.10 CMAPI_IoT_GetBack-OffTimer()

The **CMAPI_IoT_GetBack-OffTimer()** function is used to retrieve the time left of the back-off Timer.

Prototype
dword **CMAPI_IoT_GetBack-OffTimer** (dword deviceID, dword* pTimeLeft)

Parameters

Field Name	Mode	Description
deviceID	Input	The ID of the device concerned
pTimeLeft	Output	The time left of the back off timer to be able to proceed to actions such as requesting IMSI attach/GPRS attach/PDP Context Activation/SMS-MO (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.
0X00000121	The device does not offer this capability
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 previous 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, Located_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 to 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 Application 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 from Connection Manager Application being alerted twice for one notification.

Prototype
dword **CMAPI_Callback_SMS** (dword deviceID, dword systemID, dword msgID, dword mailbox, dword totalSegements, 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_WLANNewMO()

The **CMAPI_Callback_WLANNewMO()** function is used to notify that a new or an updated WLAN MO has been provided to the Terminal.

Prototype
dword **CMAPI_Callback_WLANNewMO** (dword deviceID, dword wMO)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
wMO	Input	The type of new or updated WLAN MO: • 0x00000000: Reserved • 0x00000001: HS 2.0 MO • 0x00010000: ANDSF MO

8.3.28 CMAPI_Callback_WLANSettingsChanged()

The **CMAPI_Callback_WLANSettingsChanged ()** function is used to notify that a WLAN operator setting has been changed.

Prototype
dword **CMAPI_Callback_WLANSettingsChanged** (dword deviceID, dword operatorSetting)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
operatorSetting	Input	The New operator settings regarding which WLAN policies should apply: • 0x00000000: Not specified – no preferences between ANDSF & HS2.0 • 0x00000001: ANDSF Preferred (default) • 0x00000002: HS2.0 Preferred • 0x00000003: ANDSF Only Any other value is interpreted as default value (ANDSF preferred – 0x00000001)

8.3.29 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.

8.3.30 CMAPI_Callback_OMADMStatus()

This callback indicates any OMA-DM operation Progress or Status in between. 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 successfully) • 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

8.3.31 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] and [3GPP TS 51.014] for the 3GPP specific part, in [3GPP2 C.S0035] for the 3GPP2 specific part.

8.3.32 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] and [3GPP TS 51.014] for the 3GPP specific part, in the chapter "Structure and coding of the TERMINAL PROFILE" of [3GPP2 C.S0035] for the 3GPP2 specific part.

8.3.33 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.

8.3.34 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

8.3.35 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

8.3.36 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

8.3.37 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

8.3.38 CMAPI_Callback_Incoming_Voice_Call()

The **CMAPI_Callback_Incoming_Voice_Call()** function is used to provide to application information regarding a voice call state (incoming, established...).

Prototype
dword **CMAPI_Callback_Incoming_Voice_Call** (dword deviceID, dword systemID, dword state)

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
state	Input	The status of an incoming voice call • 0x00000001: Incoming Voice Call • 0x00000002: Voice Call Established • 0x00000003: Voice Call Terminated

8.3.39 CMAPI_Callback_SEServicesChange()

The **CMAPI_Callback_SEServicesChange()** function is used to communicate changes regarding the availability of services. An updated list is provided to the application. The method used to update this list (event, trigger, polling) is out of scope of the OpenCMAPI enabler.

Prototype
dword **CMAPI_Callback_SEServicesChange** (dword deviceID, SEServices* pSEServices)

Parameters		
Field Name	**Mode**	**Description**
deviceID	Input	The ID of the device concerned
pSEServices	Input	All SE services supported by the device. See SEServices structure definition

8.3.40 CMAPI_Callback_BatteryStatusChanged()

The **CMAPI_Callback_BatteryStatusChanged()** function is used to communicate whenever there is a change in the battery status.

Prototype
dword **CMAPI_Callback_BatteryStatusChanged** (dword deviceID, dword powerSourceState, dword charging State, dword batteryLevel, dword remainingTime)

Parameters

Field Name	Mode	Description
deviceID	Input	The ID of the device concerned
powerSourceState	Input	The Indication of the power source state • 0x00000000: External Source • 0x00000001: Battery
chargingState	Input	The Indication of the charging state • 0x00000000: Charging • 0x00000001: Not charging
batteryLevel	Input	Battery level in percentage
remaningTime	Input	(Optional) Estimated Remaining time in the battery in minutes 0xFFFFFFFF not provided

8.3.41 CMAPI_Callback_BatteryThresholdReached()

The **CMAPI_Callback_BatteryThresholdReached()** function is used to communicate whenever the battery level of the device is reaching a threshold set by the function **CMAPI_Information_SetBatteryThreshold()**.

Prototype
dword **CMAPI_Callback_BatteryThresholdReached** (dword deviceID, dword batteryLevel, dword remainingTime)

Parameters

Field Name	Mode	Description
deviceID	Input	The ID of the device concerned
batteryLevel	Input	Battery level in percentage
remaningTime	Input	(Optional) Estimated Remaining time in the battery in minutes 0xFFFFFFFF not provided

8.3.42 CMAPI_Callback_P2P_DiscoveryMatch()

The **CMAPI_Callback_P2P_DiscoveryMatch()** function is used to alert the Local Device of a DeviceID/ServiceID/ServiceRecord discovery match as indicated in **CMAPI_P2P_DiscoveryMonitor()**.

Prototype
dword **CMAPI_Callback_P2P_DiscoveryMatch** (CallbackStatus status, dword deviceID, dword remoteDeviceID, UTF8* serviceID, dword result)

Parameters		
Field Name	**Mode**	**Description**
status	Input	The status of the callback.
deviceID	Input	The ID of the device concerned.
remoteDeviceID	Input	Optional; Remote Devices that match the criteria
serviceID	Input	Service Identifiers that match the criteria
result	Input	• 0x00000001: Device & Service matching criteria • 0x00000002: Device only matching criteria • 0x00000003: Service only matching criteria

8.3.43 CMAPI_Callback_P2P_Connection()

This callback shows the result of call made to **CMAPI _P2P_EstablishConnection().**

Prototype
dword **CMAPI_P2P_Connection**(CallbackStatus status, dword deviceID, dword remoteDeviceID, dqword serviceID, dword connectionID, dword result)

Parameters		
Field Name	**Mode**	**Description**
status	Input	The status of the callback.
deviceID	Input	The ID of the device concerned.
remoteDeviceID	Input	The ID of the Remote Device concerned.
serviceID	Input	The ID of the service concerned.

connectionID	Input	The ID of the connection concerned.
result	Input	• 0x00000001: No answer • 0x00000002: Connection request accepted by remote device • 0x00000003: Connection request rejected by remote device • 0x00000004: Connection established

8.3.44 CMAPI_Callback_P2P_GroupNotification()

The **CMAPI_Callback_P2P_GroupNotification()** function is invoked as a result of a previous call to **CMAPI_P2P_JoinGroup()**.

Prototype
dword **CMAPI_Callback_P2P_GroupNotification** (CallbackStatus status, dword deviceID, dword remoteDeviceID, dqword serviceID, dword result)

Parameters		
Field Name	**Mode**	**Description**
status	Input	The status of the callback.
deviceID	Input	The ID of the device concerned.
remoteDeviceID	Input	The ID of the Remote Device concerned.
serviceID	Input	The ID of the service concerned.
result	Input	• 0x00000001: No answer • 0x00000002: Invitation accepted • 0x00000003: Invitation rejected

8.3.45 CMAPI_Callback_GetMobilitytoLocation_Complete() - Optional

This callback shows the result of a previous call made to **CMAPI_Information_GetMobilitytoLocation()**.

Prototype
dword **CMAPI_Callback_GetMobilitytoLocation_Complete** (CallbackStatus status, dword deviceID, dword state, dword speed, dword speedaccuracy)

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

status	Input	The status of the callback.
deviceID	Input	The ID of the device concerned
state	Input	The state of mobility compared to the location specified • 0x00000000: Same distance to Location (accuracy of 5 meters is allowed) • 0x00000001: Distance to location changed
speed	Input	(Optional) Estimated Speed in meters per second 0xFFFFFFFF not provided
speedaccuracy	Input	(Optional) Estimated Speed accuracy in meters per second 0xFFFFFFFF not provided

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
0X00000112	The routerID references a non-existing router
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
0X00000140	The MACAddress references a non-existing Connected Device
0X00000150	The threshold value(s) is/are invalid
0X00000151	The location is invalid
0X00000160	The PDP context ID is invalid
0X00000161	The PDP Type is invalid
0X00000164	The back off time interval 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
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, only happen when creating a profile with a 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
Contact Management Error Codes	
0X00005501	The contact record is invalid
0X00005502	Memory capacity exceeded.
0X00005503	The index is invalid
0X00005504	The contact location value 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
P2P Direct Management Error Codes	
0X00008001	The P2PTechnology is not supported
0X00008002	The P2P Technology is invalid
0X00008003	The Service Record is invalid
0X00008004	The list of Remote Devices is invalid.
0X00008005	The list of Service Identifiers is invalid.
0X00008006	The ID of the Connection is invalid.
0X00008007	The list of Device ID is invalid
0X00008008	The ID of the group is invalid
0X00008009	The ID of the Remote Device is invalid
0X0000800A	The Invitation ID is invalid
Router Management Error Codes	
0X00009001	The routerConfig value(s) are incorrect
0X00009002	The policy value(s) are incorrect
0X00009003	The restrict value(s) are incorrect
0X00009004	The administrator password is incorrect
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
0X0001001A	The User Preference is invalid
0X0001001B	The User Preference Priority is invalid
0X0001001C	The Operator Preference is invalid
0X0001001D	The Operator Preference Priority is invalid
0X00010020	The WLAN Network cannot be blacklisted (i.e. already in the operator preferred network list)
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
0X00014001	The Advertisement Protocol Element is invalid
0X00014002	The Query List ANQP element is invalid
0X00014003	The HS Query List is invalid

0X00014021	HS 2.0 MO is not supported by the device
0X00014022	ANDSF MO is not supported by the device
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.
0X30000029	The pDevAttributes Buffer is not large enough.

0X3000002A	The pANQPQResponse buffer is not large enough. pANQPQResponseSize contains the minimum buffer length required.
0X3000002B	The pHSQResponse buffer is not large enough. pHSQResponseSize contains the minimum buffer length 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 pServingNetworkInfoSize will contain the minimum number of bytes required.
0X30000032	The buffer is not sufficient to hold the data, the pConnRecordSize will contain the minimum number of bytes required.
0X30000033	The buffer is not sufficient to hold the data, the pPDPContextListSize 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.
0X30001001	The pIMPU buffer is not large enough. pIMPUSize contains the minimum buffer length required.
0X30001002	The pIARI buffer is not large enough. pIARISize contains the minimum buffer length required.
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.
0X30009001	The pRouterConfigList buffer is not large enough
0X30009002	The pConDevList buffer is not large enough
0X30009003	The pPolicyList buffer is not large enough
0X30009004	The pRestrictList 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.
0X30010004	The pSEServices Buffer is not large enough, pSEServicesSize contains the required size in bytes
0X30040021	The pHS2MOSubscription is not large enough.
0X30040022	The pAMOSubscription is not large enough.
0X30040031	The pUWSIDL buffer is not large enough. pUWSIDLSize contains the minimum

	buffer length required.
0X30040032	The pOWSIDL buffer is not large enough. pOWSIDLSize contains the minimum buffer length required.
M2M/IoT related Error Codes	
	List of errors related to M2M/IoT
0X4001XXXX	Operation cannot be done – Back off timer in place – time left is indicated by the 4 last digits (in seconds) XXXX is the time left in seconds - example 0X40010360 means 360 seconds are left
0X401MMCME	Error codes related to GSM Mobility Management where MM indicates the Mobility Management Cause code and CME the code for Mobile Equipment error.
0X402GMCME	Error codes related to GPRS Mobility Management where GM indicates the GPRS Mobility Management Cause code and CME the code for Mobile Equipment error.
0X403SMCME	Error codes related to Session Management where SM indicates the Session Management Cause code and CME the code for Mobile Equipment error.
0X404XXCMS	Error codes related to other reasons where XX indicates other Cause code and CMS the code for Mobile Equipment specific error.
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
0XF0000006	The WLAN Encryption Type used is not allowed. Please use proper Encryption type

Table 8: 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 9: Status Words Codes

9.3 CMEE codes

The following tables are listing possible GSM Mobile Equipment error codes and GSM network error codes.

Causes Codes Related to GSM Mobility Management				
MM Code	**CME Code**	**Cause**	**Reason**	**Action proposed**
2		IMSI unknown in HLR	This cause is sent to the MS if the MS is not known (registered) in the HLR. This cause code does not affect operation of the GPRS service, although it may be used by a GMM procedure.	
3	103	Illegal MS	This cause is sent to the MS when the network refuses service to the MS either because an identity of the MS is	

			not acceptable to the network or because the MS does not pass the authentication check, i.e. the SRES received from the MS is different from that generated by the network.	
4		IMSI unknown in VLR	This cause is sent to the MS when the given IMSI is not known at the VLR.	
5		IMEI not accepted	This cause is sent to the MS if the network does not accept emergency call establishment using an IMEI.	
6	106	Illegal ME	This cause is sent to the MS if the ME used is not acceptable to the network, e.g. blacklisted.	
11	111	PLMN not allowed	This cause is sent to the MS if it requests location updating in a PLMN where the MS, by subscription or due to operator determined barring is not allowed to operate.	
12	112	Location Area not allowed	This cause is sent to the MS if it requests location updating in a location area where the MS, by subscription, is not allowed to operate.	
13	113	Roaming not allowed in this location area	This cause is sent to an MS which requests location updating in a location area of a PLMN which restricts roaming to that MS in that Location Area, by subscription.	
17	615	Network failure	This cause is sent to the MS if the MSC cannot service an MS generated request because of PLMN failures, e.g. problems in MAP.	Additionally, retry retries may be attempted, but no more frequently than once every 60 seconds
22	42	Congestion	This cause is sent if the service request cannot be processed because of congestion (e.g. no channel, facility busy/congested etc.)	Additionally, retry retries may be attempted, but no more frequently than once every 60 seconds
32	132	Service Option Not Supported.	This cause is sent when the MS requests a service/facility in the CM SERVICE REQUEST message which is not supported by the PLMN.	Additionally, device should not retry the attempt on the same PLMN unless prompted externally to do so (i.e. modem should not automatically retry).
33	133	Requested Service Option Not Subscribed	This cause is sent when the MS requests a service option for which it has no subscription.	Additionally, device should not retry the attempt unless prompted externally to do so (i.e. modem should not automatically retry).
34	134	Service option temporarily out of	This cause is sent when the MSC cannot service the request because of	Additionally, retry retries may be attempted, but no more

		order	temporary outage of one or more functions required for supporting the service.	frequently than once every 60 seconds
38		Call Cannot be identified	This cause is sent when the network cannot identify the call associated with a call re-establishment request.	
Causes Codes Related to GPRS Mobility Management				
GM Code	**CME Code**	**Cause**	**Reason**	**Action proposed**
7	107	GPRS Services Not Allowed	This cause is sent to the MS if it requests an IMSI attach for GPRS services, but is not allowed to operate GPRS services.	
8		GPRS services and non-GPRS services not allowed	This cause is sent to the MS if it requests a combined IMSI attach for GPRS and non-GPRS services, but is not allowed to operate either of them.	
9		MS identity cannot be derived by the network	This cause is sent to the MS when the network cannot derive the MS's identity from the P-TMSI in case of inter-SGSN routing area update.	
10		Implicitly detached	This cause is sent to the MS either if the network has implicitly detached the MS, e.g. some while after the Mobile reachable timer has expired, or if the GMM context data related to the subscription does not exist in the SGSN e.g. because of a SGSN restart.	
14	111	GPRS services not allowed in this PLMN	This cause is sent to the MS which requests GPRS service in a PLMN which does not offer roaming for GPRS services to that MS.	
16		MSC temporarily not reachable	This cause is sent to the MS if it requests a combined GPRS attach or routing are updating in a PLMN where the MSC is temporarily not reachable via the GPRS part of the GSM network.	
	148	unspecified GPRS error		Additionally, retry retries may be attempted, but no more frequently than once every 60 seconds
Causes Codes Related to Session Management				
SM Code	**CME Code**	**Cause**	**Reason**	**Action proposed**

25		LLC or SNDCP failure	This cause code is used by the MS indicate that a PDP Context is deactivated because of a LLC or SNDCP failure (e.g. if the SM receives a SNSM-STATUS.request message with cause "DM received " or " invalid XID response)	Additionally, retry retries may be attempted, but no more frequently than once every 60 seconds
26		Insufficient resources	This cause code is used by the MS or by the network to indicate that a PDP Context activation request or PDP Context modification request cannot be accepted due to insufficient resources.	Additionally, retry retries may be attempted, but no more frequently than once every 60 seconds
27	134	Unknown or missing access point name	This cause code is used by the network to indicate that the requested service was rejected by the external packet data network because the access point name was not included although required or if the access point name could not be resolved.	Additionally, do not retry with same APN unless device is power cycled.
28		Unknown PDP address or PDP type	This cause code is used by the network to indicate that the requested service was rejected by the external packet data network because the PDP address or type could not be recognised.	Additionally, do not retry with same PDP address and/or type unless device is power cycled.
29	149	User authentication failed	This cause code is used by the network to indicate that the requested service was rejected by the external packet data network due to a failed user authentication (e.g. rejected by Radius)	Additionally, retry retries may be attempted, but no more frequently than once every 60 seconds
30		Activation rejected by GGSN	This cause code is used by the network to indicate that the requested service was rejected by the GGSN.	Additionally, retry retries may be attempted, but no more frequently than once every 60 seconds
31		Activation rejected, unspecified	This cause code is used by the network to indicate that the requested service was rejected due to unspecified reasons.	Additionally, retry retries may be attempted, but no more frequently than once every 60 seconds
32	132	Service option not supported	This cause code is used by the network when the MS requests a service which is not supported by the PLMN.	Additionally, device should not retry the attempt on the same PLMN unless prompted externally to do so (i.e. modem should not automatically retry).
33	133	Requested service option not subscribed	This cause is sent when the MS requests a service option for which it has no subscription.	Additionally, device should not retry the attempt on the same PLMN unless prompted externally to do so (i.e. modem should not

				automatically retry).
34	134	Service option temporarily out of order	This cause is sent when the MSC cannot service the request because of temporary outage of one or more functions required for supporting the service.	Additionally, retry retries may be attempted, but no more frequently than once every 60 seconds
35		NSAPI already used	This cause code is used by the network to indicate that the NSAPI requested by the MS in the PDP Context activation is already used by another active PDP Context of this MS.	Device may choose to use a different NSAPI, or retry after the context using the required NSAPI has been deactivated.
36		Regular PDP Context deactivation	This cause code is used to indicate a regular MS or network initiated PDP Context deactivation.	
37		QoS not accepted	This cause code is used by the MS if the new QoS cannot be accepted that were indicated by the network in the PDP Context Modification procedure.	N/A
38	615	Network Failure	This cause code is used by the network to indicate that the PDP Context deactivation is caused by an error situation in the network.	Additionally, retry retries may be attempted, but no more frequently than once every 60 seconds
39		Reactivation requested	This cause code is used by the network to request a PDP Context reactivation after a GGSN restart.	Additionally, the device may re-establish the PDP Context.
40		Feature not supported	This cause code is used by the MS to indicate that the PDP Context activation initiated by the network is not supported by the MS.	N/A
Causes Codes Related to other reasons				
XX Code	**CMS Code**	**Cause**	**Reason**	**Action proposed**
	8	Operator determined barring	This cause indicates that the device has tried to send a mobile originating short message when the device's network operator or service provider has forbidden such transactions.	SMS back-off, blocking immediately any new SMS TX request sent
	10	Call barred	This cause indicates that the outgoing call barred service applies to the short message service for the called destination.	SMS back-off, blocking immediately any new SMS TX request sent
	21	Short message transfer rejected	This cause indicates that the equipment sending this cause does not wish to accept this short message, although it could have accepted the short message since the equipment sending this cause is neither busy nor	SMS back-off, blocking immediately any new SMS TX request sent

			incompatible.	
	27	Destination out of service	This cause indicates that the destination indicated by the Device cannot be reached because the interface to the destination is not functioning correctly. The term "not functioning correctly" indicates that a signalling message was unable to be delivered to the remote user; e.g., a physical layer or data link layer failure at the remote user, user equipment off-line, etc.	SMS back-off, blocking immediately any new SMS TX request sent
	28	Unidentified subscriber	This cause indicates that the subscriber is not registered in the PLMN (i.e. IMSI not known).	SMS back-off, blocking immediately any new SMS TX request sent
	29	Facility rejected	This cause indicates that the facility requested by the Device is not supported by the PLMN.	SMS back-off, blocking immediately any new SMS TX request sent
	30	Unknown subscriber	This cause indicates that the subscriber is not registered in the HLR (i.e. IMSI or directory number is not allocated to a subscriber).	SMS back-off, blocking immediately any new SMS TX request sent
	38	Network out of order	This cause indicates that the network is not functioning correctly and that the condition is likely to last a relatively long period of time; e.g., immediately reattempting the short message transfer is not likely to be successful.	SMS back-off, blocking immediately any new SMS TX request sent
	41	Temporary failure	This cause indicates that the network is not functioning correctly and that the condition is not likely to last a long period of time; e.g., the Device may wish to try another short message transfer attempt almost immediately.	SMS back-off, blocking immediately any new SMS TX request sent
	42	Congestion	This cause indicates that the short message service cannot be serviced because of high traffic.	SMS back-off, blocking immediately any new SMS TX request sent
	47	Resources unavailable, unspecified	This cause is used to report a resource unavailable event only when no other cause applies.	SMS back-off, blocking immediately any new SMS TX request sent
	50	Requested facility not subscribed	This cause indicates that the requested short message service could not be provided by the network because the user has not completed the necessary administrative arrangements with its supporting networks.	SMS back-off, blocking immediately any new SMS TX request sent
	69	Requested facility	This cause indicates that the network is unable to provide the requested	SMS back-off, blocking immediately any new SMS

		not implemented	short message service.	TX request sent
	81	Invalid short message transfer reference value	This cause indicates that the equipment sending this cause has received a message with a short message reference which is not currently in use on the MS-network interface.	SMS back-off, blocking immediately any new SMS TX request sent
	148	Unspecified GPRS error		
17		Network failure	This cause is sent to the MS if the MSC cannot service an MS generated request because of PLMN failures, e.g. Problems in MAP.	SMS back-off, blocking immediately any new SMS TX request sent
21		Congestion	This cause is sent if the service request cannot be processed because of congestion (e.g. no channel, facility busy/congested, etc).	SMS back-off, blocking immediately any new SMS TX request sent

Table 10: CMEE Codes

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
n/a	n/a	No prior version

A.2 Draft/Candidate Version 1.1 History

Document Identifier	Date	Sections	Description
Draft Versions OMA-TS-OpenCMAPI-V1_1	18 Feb 2013	Baseline	Incorporates the baseline TS v1.1 based on the TS version 1.0: OMA-CD-OpenCMAPI-2013-0024R01-INP_TS_1.1_baseline
	25 Apr 2013	6,7,8	Incorporated the following CR: OMA-CD-OpenCMAPI-2013-0032R02-CR_TS_Contact_Mgt
	22 May 2013	All	Incorporated the following CRs: OMA-CD-OpenCMAPI-2013-0042R04-CR_TS_WebAPI_JSON OMA-CD-OpenCMAPI-2013-0045R01-CR_Callback_Incoming_Voice_Call
	01 Jul 2013	All	Incorporated the following CRs: OMA-CD-OpenCMAPI-2013-0046-CR_TS_P2P OMA-CD-OpenCMAPI-2013-0048R03-CR_Extended_capabilities_NFC OMA-CD-OpenCMAPI-2013-0051R02-CR_Device_capabilities
	24 Sep 2013	All	Incorporated the following CRs: OMA-CD-OpenCMAPI-2013-0047R01-CR_TS_Search_Contacts OMA-CD-OpenCMAPI-2013-0052R03-CR_Extended_capabilities_ESE OMA-CD-OpenCMAPI-2013-0066R03-CR_TS_WirelessRouters OMA-CD-OpenCMAPI-2013-0068R03-CR_TS_Battery_Status OMA-CD-OpenCMAPI-2013-0073R02-CR_P2P_Direct_Mgt_APIs_part_2 OMA-CD-OpenCMAPI-2013-0075-CR_TS_WebAPI_Change OMA-CD-OpenCMAPI-2013-0090R01-CR_TS_Functions_supported
	21 Nov 2013	All	Incorporated the following CRs: OMA-CD-OpenCMAPI-2013-0050R02-CR_TS_SMS OMA-CD-OpenCMAPI-2013-0091R01-CR_TS_Profile_Management. OMA-CD-OpenCMAPI-2013-0092-CR_TS_WLAN_Changes OMA-CD-OpenCMAPI-2013-0100R01-CR_TS_Bug_NFCInfoType_GetSignalStrength OMA-CD-OpenCMAPI-2013-0102-CR_v1_0_fixes OMA-CD-OpenCMAPI-2013-0123-CR_P2P_Direct_Mgt_APIs_part_3 OMA-CD-OpenCMAPI-2013-0127-CR_New_fixes_from_v1_0 OMA-CD-OpenCMAPI-2013-0134-CR_N_fixes_from_v1_0 OMA-CD-OpenCMAPI-2013-0143-CR_New_Edit_fixes_from_v1_0. OMA-CD-OpenCMAPI-2013-0145-CR_More_bug_fixes_from_v1_0. OMA-CD-OpenCMAPI-2013-0145R01-CR_More_bug_fixes_from_v1_0. OMA-CD-OpenCMAPI-2013-0149-CR_Add_bug_fixes_from_v1_0. OMA-CD-OpenCMAPI-2013-0151R01-CR_Pointer_fixes_v1.1. Editorial changes
	12 Dec 2013	5, 6, 7, 8	Incorporated the following CRs: OMA-CD-OpenCMAPI-2013-0159R01-CR_encore_bug_fixes_v1_1
	25 Dec 2013	7, 8, 9	Incorporated the following CRs: OMA-CD-OpenCMAPI-2013-0164-CR_Bug_fixes_pointers OMA-CD-OpenCMAPI-2013-0170-CR_Add_Error_codes
	27 Jan 2014	7 & Appendix B	Added Editorial notes in accordance with OpenCMAPI-2014-A002 Incorporated the following CRs: OMA-CD-OpenCMAPI-2014-0005-CR_TS_SCR_update
	18 Feb 2014	All	Updated according to resolution of CONRR comments closed or addressed on 20140218: C003, C004, C005, C006, C010, C013, C015, C017, C019, C021, C024, C025, C026, C027, C028, C029

Document Identifier	Date	Sections	Description
	20 Feb 2014	All	Updated according to resolution of CONRR comments closed or addressed on 20140220: C002, C007, C008, C012
	08 Apr 2014	6, 7, 8, 9	Small Editorial Changes Incorporated the following CRs: OMA-CD-OpenCMAPI-2014-0014-CR_ProfileID_v1_1
	22 Apr 2014	6, 7, 8	Small Editorial Changes Updated according to resolution of CONRR comment closed C003 Incorporated the following CRs: OMA-CD-OpenCMAPI-2014-0026-CR_TS_resolution_C003
	02 Aug 2014	All	Incorporated: OMA-CD-OpenCMAPI-2014-0030-CR_CONR_A059_resolution OMA-CD-OpenCMAPI-2014-0034-CR_TS_CONR_Resolution_C030 OMA-CD-OpenCMAPI-2014-0046R01-CR_Resolution_of_P2P_related_comments_to_TS OMA-CD-OpenCMAPI-2014-0053R01-CR_TS_CONR_ConnectionRecords OMA-CD-OpenCMAPI-2014-0052R01-CR_TS_CONR_MobilityState OMA-CD-OpenCMAPI-2014-0062-CR_TS_CONR_Resolution_C018 OMA-CD-OpenCMAPI-2014-0065-CR_TS_CONR_ISIM OMA-CD-OpenCMAPI-2014-0035R012-CR_TS_CONR_Resolution_WLAN OMA-CD-OpenCMAPI-2014-0038R01-CR_TS_CONR_Resolution_WLAN_part2 OMA-CD-OpenCMAPI-2014-0039R01-CR_TS_CONR_Resolution_WLAN_part3 OMA-CD-OpenCMAPI-2014-0040R01-CR_TS_CONR_Resolution_WLAN_part3 OMA-CD-OpenCMAPI-2014-0041-CR_TS_CONR_Resolution_WLAN_part5 OMA-CD-OpenCMAPI-2014-0048R01-CR_TS_CONR_Resolution_WLAN_part6 OMA-CD-OpenCMAPI-2014-0055R01-CR_TS_CONR_Resolution_WLAN_part7 OMA-CD-OpenCMAPI-2014-0058-CR_TS_CONR_Resolution_WLAN_part8 OMA-CD-OpenCMAPI-2014-0059R01-CR_TS_CONR_Resolution_WLAN_part9
	15 Sep 2014	All	Incorporated: OMA-CD-OpenCMAPI-2014-0061R01-CR_TS_CONR_Resolution_C009 OMA-CD-OpenCMAPI-2014-0083-CR_TS_bugfixes OMA-CD-OpenCMAPI-2014-0084R01-CR_TS_Update_WLAN OMA-CD-OpenCMAPI-2014-0085-CR_TS_IoT_APIs OMA-CD-OpenCMAPI-2014-0086-CR_TS_AppendixC_Update OMA-CD-OpenCMAPI-2014-0087-CR_TS_Callbacks_Update OMA-CD-OpenCMAPI-2014-0091-CR_TS_Update_IMPU
	16 Sep 2014	All	Updated according to resolution of CONRR comments closed or addressed on 20140916: C001, C011, C019 Incorporated: OMA-CD-OpenCMAPI-2014-0093-CR_TS_Editorial_Update OMA-CD-OpenCMAPI-2014-0094-CR_TS_ANQP_fixes
	30 Sep 2014	All	Small Editorial changes
Candidate Version OMA-TS-OpenCMAPI-V1_1	17 Feb 2015	n/a	Status changed to Candidate by TP TP Ref # OMA-TP-2015-0059-INP_OpenCMAPI_V1_1_ERP_and_ETR_for_Candidate_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	7.10	
OpenCMAPI-MBD-009-O	Support UICC Management APIs	7.11	
OpenCMAPI-MBD-010-O	Support WLAN APIs	7.12	
OpenCMAPI-MBD-011-M	Support Statistics APIs	7.13	
OpenCMAPI-MBD-012-M	Support Information Status APIs	7.14	
OpenCMAPI-MBD-013-M	Support SMS Management APIs	7.15	
OpenCMAPI-MBD-014-M	Support USSD Management APIs	7.16	
OpenCMAPI-MBD-015-O	Support GNSS APIs	7.17	
OpenCMAPI-MBD-016-O	Support Data Push Service Management APIs	7.18	
OpenCMAPI-MBD-017-M	Support Callback APIs	8	
OpenCMAPI-MBD-018-O	Support Device Extended Service APIs	7.9	
OpenCMAPI-MBD-019-M	Support Contact Management APIs	7.19	
OpenCMAPI-MBD-020-O	Support P2P Direct Management APIs	7.20	
OpenCMAPI-MBD-021-O	Support Wireless Router APIs	7.21	
OpenCMAPI-MBD-022-O	Support IP Multimedia Services APIs	7.22	
OpenCMAPI-MBD-023-O	Support M2M/IoT APIs	7.23	

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	7.10	
OpenCMAPI-LAP-009-O	Support UICC Management APIs	7.11	

Item	Function	Reference	Requirement
OpenCMAPI-LAP-010-M	Support WLAN APIs	7.12	
OpenCMAPI-LAP-011-M	Support Statistics APIs	7.13	
OpenCMAPI-LAP-012-M	Support Information Status APIs	7.14	
OpenCMAPI-LAP-013-M	Support SMS Management APIs	7.15	
OpenCMAPI-LAP-014-M	Support USSD Management APIs	7.16	
OpenCMAPI-LAP-015-O	Support GNSS APIs	7.17	
OpenCMAPI-LAP-016-O	Support Data Push Service Management APIs	7.18	
OpenCMAPI-LAP-017-M	Support Callback APIs	8	
OpenCMAPI-LAP-018-O	Support Device Extended Service APIs	7.9	
OpenCMAPI-LAP-019-M	Support Contact Management APIs	7.19	
OpenCMAPI-LAP-020-O	Support P2P Direct Management APIs	7.20	
OpenCMAPI-LAP-021-O	Support Wireless Router APIs	7.21	
OpenCMAPI-LAP-022-O	Support IP Multimedia Services APIs	7.22	
OpenCMAPI-LAP-023-O	Support M2M/IoT APIs	7.23	

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	7.10	
OpenCMAPI-WIR-009-O	Support UICC Management APIs	7.11	
OpenCMAPI-WIR-010-O	Support WLAN APIs	7.12	
OpenCMAPI-WIR-011-M	Support Statistics APIs	7.13	
OpenCMAPI-WIR-012-M	Support Information Status APIs	7.14	
OpenCMAPI-WIR-013-M	Support SMS Management APIs	7.15	
OpenCMAPI-WIR-014-O	Support USSD Management APIs	7.16	
OpenCMAPI-WIR-015-O	Support GNSS APIs	7.17	
OpenCMAPI-WIR-016-O	Support Data Push Service Management APIs	7.18	
OpenCMAPI-WIR-017-M	Support Callback APIs	8	
OpenCMAPI-WIR-018-O	Support Device Extended Service APIs	7.9	
OpenCMAPI-WIR-019-M	Support Contact Management APIs	7.19	
OpenCMAPI-WIR-020-O	Support P2P Direct Management APIs	7.20	
OpenCMAPI-WIR-021-M	Support Wireless Router APIs	7.21	
OpenCMAPI-WIR-022-O	Support IP Multimedia Services APIs	7.22	
OpenCMAPI-WIR-023-O	Support M2M/IoT APIs	7.23	

B.4 SCR for M2M device

B.4.1 General M2M device

Item	Function	Reference	Requirement
OpenCMAPI-M2M-001-M	Support API Management	7.2	

Item	Function	Reference	Requirement
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	7.10	
OpenCMAPI-M2M-009-O	Support UICC Management APIs	7.11	
OpenCMAPI-M2M-010-O	Support WLAN APIs	7.12	
OpenCMAPI-M2M-011-M	Support Statistics APIs	7.13	
OpenCMAPI-M2M-012-M	Support Information Status APIs	7.14	
OpenCMAPI-M2M-013-M	Support SMS Management APIs	7.15	
OpenCMAPI-M2M-014-O	Support USSD Management APIs	7.16	
OpenCMAPI-M2M-015-O	Support GNSS APIs	7.17	
OpenCMAPI-M2M-016-O	Support Data Push Service Management APIs	7.18	
OpenCMAPI-M2M-017-M	Support Callback APIs	8	
OpenCMAPI-M2M-018-O	Support Device Extended Service APIs	7.9	
OpenCMAPI-M2M-019-O	Support Contact Management APIs	7.19	
OpenCMAPI-M2M-020-O	Support P2P Direct Management APIs	7.20	
OpenCMAPI-M2M-021-O	Support Wireless Router APIs	7.21	
OpenCMAPI-M2M-022-O	Support IP Multimedia Services APIs	7.22	
OpenCMAPI-M2M-023-O	Support M2M/IoT APIs	7.23	

B.4.2 Basic M2M device

Basic M2M device is a subset of M2M device representing devices that are able to perform only basic functions such as a sensor or a meter. These basic M2M devices are also referred as IoT (Internet of Things) devices.

Therefore, for each group of requirements, only some functions will be supported by Basic M2M devices (only the Mandatory functions are listed here – Any function not mentioned below is considered as Optional for Basic M2M).

Item	Function	Reference	Requirement
OpenCMAPI-IoT-001-M	CMAPI_API_Open()	7.2	
OpenCMAPI-IoT-002-M	CMAPI_API_Close()	7.2	
OpenCMAPI-IoT-003-M	CMAPI_API_GetOpenCMAPIVersion()	7.2	
OpenCMAPI-IoT-004-M	CMAPI_API_GetFunctionsSupported()	7.2	
OpenCMAPI-IoT-005-M	CMAPI_Discovery_OpenDevice()	7.3	
OpenCMAPI-IoT-006-M	CMAPI_Discovery_CloseDevice()	7.3	
OpenCMAPI-IoT-007-M	CMAPI_Network_GetRFInfo()	7.4	
OpenCMAPI-IoT-008-M	CMAPI_NetCon_GetConnectionStatus()	7.6	
OpenCMAPI-IoT-009-M	CMAPI_NetCon_SetAutoConnectMode()	7.6	
OpenCMAPI-IoT-010-M	CMAPI_NetCon_GetAutoConnectMode()	7.6	
OpenCMAPI-IoT-011-M	CMAPI_NetCon_SetPermittedBearers()	7.6	
OpenCMAPI-IoT-012-M	CMAPI_NetCon_GetPermittedBearers()	7.6	
OpenCMAPI-IoT-013-M	CMAPI_DevSrv_GetIMSI()	7.8	
OpenCMAPI-IoT-014-M	CMAPI_DevSrv_GetDeviceStatus()	7.8	
OpenCMAPI-IoT-015-M	CMAPI_DevSrv_GetFirmwareVersion()	7.8	
OpenCMAPI-IoT-016-M	CMAPI_DevSrv_GetRFSwitch()	7.8	
OpenCMAPI-IoT-017-M	CMAPI_DevSrv_SetRadioState()	7.8	

Item	Function	Reference	Requirement
OpenCMAPI-IoT-018-M	CMAPI_Information_GetNetworkSelectionMode()	7.14	
OpenCMAPI-IoT-019-M	CMAPI_Information_GetSignalStrength()	7.14	
OpenCMAPI-IoT-020-M	CMAPI_Information_GetRoamingStatus()	7.14	
OpenCMAPI-IoT-021-M	CMAPI_Information_GetRATType()	7.14	
OpenCMAPI-IoT-022-M	CMAPI_Information_GetRadioState()	7.14	
OpenCMAPI-IoT-023-M	CMAPI_Information_GetBatteryStatus()	7.14	
OpenCMAPI-IoT-024-M	CMAPI_SMS_Send()	7.15	
OpenCMAPI-IoT-025-O	CMAPI_IoT_IMSI_Attach()	7.23	Optional function but recommended for this type of device
OpenCMAPI-IoT-026-O	CMAPI_IoT_GPRS_Register()	7.23	Optional function but recommended for this type of device
OpenCMAPI-IoT-027-O	CMAPI_IoT_Set_PDPContext()	7.23	Optional function but recommended for this type of device
OpenCMAPI-IoT-028-O	CMAPI_IoT_GetPDPContextList()	7.23	Optional function but recommended for this type of device
OpenCMAPI-IoT-029-O	CMAPI_IoT_GetPDPContextIPaddress()	7.23	Optional function but recommended for this type of device
OpenCMAPI-IoT-030-O	CMAPI_IoT_Activate_PDPContext()	7.23	Optional function but recommended for this type of device
OpenCMAPI-IoT-031-O	CMAPI_IoT_SetNFM()	7.23	Optional function but recommended for this type of device
OpenCMAPI-IoT-032-O	CMAPI_IoT_GetNFM()	7.23	Optional function but recommended for this type of device
OpenCMAPI-IoT-033-O	CMAPI_IoT_SetBack-OffBaseInterval()	7.23	Optional function but recommended for this type of device
OpenCMAPI-IoT-034-O	CMAPI_IoT_GetBack-OffTimer()	7.23	Optional function but recommended for this type of device

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	7.10	

Item	Function	Reference	Requirement
OpenCMAPI-SMA-009-M	Support UICC Management APIs	7.11	
OpenCMAPI-SMA-010-M	Support WLAN APIs	7.12	
OpenCMAPI-SMA-011-M	Support Statistics APIs	7.13	
OpenCMAPI-SMA-012-M	Support Information Status APIs	7.14	
OpenCMAPI-SMA-013-M	Support SMS Management APIs	7.15	
OpenCMAPI-SMA-014-M	Support USSD Management APIs	7.16	
OpenCMAPI-SMA-015-O	Support GNSS APIs	7.17	
OpenCMAPI-SMA-016-M	Support Data Push Service Management APIs	7.18	
OpenCMAPI-SMA-017-M	Support Callback APIs	8	
OpenCMAPI-SMA-018-O	Support Device Extended Service APIs	7.9	
OpenCMAPI-SMA-019-M	Support Contact Management APIs	7.19	
OpenCMAPI-SMA-020-O	Support P2P Direct Management APIs	7.20	
OpenCMAPI-SMA-021-O	Support Wireless Router APIs	7.21	
OpenCMAPI-SMA-022-O	Support IP Multimedia Services APIs	7.22	
OpenCMAPI-SMA-023-O	Support M2M/IoT APIs	7.23	

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	7.10	
OpenCMAPI-TAB-009-M	Support UICC Management APIs	7.11	
OpenCMAPI-TAB-010-M	Support WLAN APIs	7.12	
OpenCMAPI-TAB-011-M	Support Statistics APIs	7.13	
OpenCMAPI-TAB-012-M	Support Information Status APIs	7.14	
OpenCMAPI-TAB-013-M	Support SMS Management APIs	7.15	
OpenCMAPI-TAB-014-M	Support USSD Management APIs	7.16	
OpenCMAPI-TAB-015-O	Support GNSS APIs	7.17	
OpenCMAPI-TAB-016-M	Support Data Push Service Management APIs	7.18	
OpenCMAPI-TAB-017-M	Support Callback APIs	8	
OpenCMAPI-TAB-018-O	Support Device Extended Service APIs	7.9	
OpenCMAPI-TAB-019-M	Support Contact Management APIs	7.19	
OpenCMAPI-TAB-020-O	Support P2P Direct Management APIs	7.20	
OpenCMAPI-TAB-021-O	Support Wireless Router APIs	7.21	
OpenCMAPI-TAB-022-O	Support IP Multimedia Services APIs	7.22	
OpenCMAPI-TAB-023-O	Support M2M/IoT APIs	7.23	

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	

Item	Function	Reference	Requirement
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	7.10	
OpenCMAPI-CLD-009-M	Support UICC Management APIs	7.11	
OpenCMAPI-CLD-010-M	Support WLAN APIs	7.12	
OpenCMAPI-CLD-011-M	Support Statistics APIs	7.13	
OpenCMAPI-CLD-012-M	Support Information Status APIs	7.14	
OpenCMAPI-CLD-013-M	Support SMS Management APIs	7.15	
OpenCMAPI-CLD-014-M	Support USSD Management APIs	7.16	
OpenCMAPI-CLD-015-O	Support GNSS APIs	7.17	
OpenCMAPI-CLD-016-O	Support Data Push Service Management APIs	7.18	
OpenCMAPI-CLD-017-M	Support Callback APIs	8	
OpenCMAPI-CLD-018-O	Support Device Extended Service APIs	7.9	
OpenCMAPI-CLD-019-M	Support Contact Management APIs	7.19	
OpenCMAPI-CLD-020-O	Support P2P Direct Management APIs	7.20	
OpenCMAPI-CLD-021-O	Support Wireless Router APIs	7.21	
OpenCMAPI-CLD-022-O	Support IP Multimedia Services APIs	7.22	
OpenCMAPI-CLD-023-O	Support M2M/IoT APIs	7.23	

Appendix C. Description of OpenCMAPI functions (Informative)

This appendix provides a list of all OpenCMAPI Functions as well as a short description and in which version they have been created.

C.1 CMAPI-1 Functions

CMAPI-1		
Function	**Description**	**Vers.**
API MANAGEMENT		
CMAPI_API_Open()	initialize the OpenCMAPI	1.0
CMAPI_API_Close()	deallocate any internal API structures including the security context	1.0
CMAPI_API_GetOpenCMAPIVersion()	retrieve the version number of the OpenCMAPI used	1.0
CMAPI_API_GetFunctionsSupported()	retrieve the OpenCMAPI groups of functions supported by the enabler	1.1
DEVICE DISCOVERY APIs		
CMAPI_Discovery_DetectDevices()	search for devices	1.0
CMAPI_Discovery_GetDevice()	discover information about the devices within the system	1.0
CMAPI_Discovery_OpenDevice()	“open” a device within the system	1.0
CMAPI_Discovery_CloseDevice()	“close” a device within the system	1.0
CELLULAR NETWORK MANAGEMENT APIs		
CMAPI_Network_GetRFInfo()	get information about RF (Radio access technology, band class, data rate supported and channel)	1.0
CMAPI_Network_GetHomeInformation()	get information about home network of the subscriber for a dedicated System	1.0
CMAPI_Network_GetServingInformation()	get information about serving network of the subscriber	1.0
CONNECTION MANAGEMENT APIs		
CMAPI_NetConnectSrv_MgrCellularProfile()	manage cellular profiles, including add/delete/update a profile information	1.0
CMAPI_NetConnectSrv_GetCellularProfile()	get the details of a specific Cellular Profile	1.0
CMAPI_NetConnectSrv_GetCellularProfileList()	get a list of all Cellular Profile names	1.0
CMAPI_NetConnectSrv_SelectNetwork()	select the current network mode and PLMN for a given System	1.0
CMAPI_NetConnectSrv_GetNetworkList_Sync()	search and compile a list of available Networks	1.0
CMAPI_NetConnectSrv_GetNetworkList_Async()	initiate the search of the Network list	1.0
CMAPI_NetConnectSrv_GetCurrentConnType()	get the current connection type	1.0
CMAPI_NetConnectSrv_Connect_Async()	connect to a network	1.0
CMAPI_NetConnectSrv_Disconnect_Async()	disconnect from the network	1.0
CMAPI_NetConnectSrv_CancelConnect_Async()	cancel of connect operation (as a result of a call to **CMAPI_NetConnectSrv_Connect_Async**)	1.0
CMAPI_NetConnectSrv_SecondaryPDPContext_Connect_Async()	connect to a network	1.0
CMAPI_NetConnectSrv_SecondaryPDPContext_Disconnect_Async()	disconnect from the network	1.0
CMAPI_NetConnectSrv_SecondaryPDPContext_CancelConnect_Async ()	cancel of connect operation (as a result of a call to **CMAPI_NetConnectSrv_SecondaryPDPContext_Connect_Async**)	1.0
NETWORK MANAGEMENT APIs		
CMAPI_NetCon_GetConnectionStatus()	obtain information about the connection status	1.0
CMAPI_NetCon_SetAutoConnectMode()	set/disable “autoconnect” mode	1.0
CMAPI_NetCon_GetAutoConnectMode()	return the current “autoconnect” mode	1.0
CMAPI_NetCon_SetDefaultProfile()	identify the profile that shall be used when the device is in auto connect mode	1.0
CMAPI_NetCon_SetPermittedBearers()	restrict the permitted mobile bearer when connecting to the selected network	1.0
CMAPI_NetCon_GetPermittedBearers()	get the current permitted bearers	1.0
CMAPI_NetCon_SetNoDataProfile()	set up (enable or disable) the nodataprofile	1.0

CMAPI_NetCon_GetNoDataProfile()	return the current state of the nodata profile (enabled or disabled)	1.0
CDMA2000 APIs		
CMAPI_CDMA2000_SetACCOLC()	set the Access Overload Class (ACCOLC) for CDMA2000 devices	1.0
CMAPI_CDMA2000_GetACCOLC()	retrieve the current value of the Access Overload Class (ACCOLC) for CDMA2000 devices	1.0
CMAPI_CDMA2000_SetCDMANetworkParameters()	set the values of certain CDMA2000-specific network parameters	1.0
CMAPI_CDMA2000_GetCDMANetworkParameters()	retrieve the values of certain CDMA2000-specific network parameters	1.0
CMAPI_CDMA2000_GetANAAAAAuthenticationStatus()	retrieve the value of the most recent ANA AAA authentication attempt status for CDMA2000 devices	1.0
CMAPI_CDMA2000_GetPRLVersion()	retrieve the value of the Preferred Roaming List (PRL) version in use for CDMA2000 devices	1.0
CMAPI_CDMA2000_GetERIFile()	retrieve the contents of the Enhanced Roaming Indicator (ERI) file in use for CDMA2000 devices	1.0
CMAPI_CDMA2000_ActivateAutomatic()	command the device to perform automatic activation using a specified activation code	1.0
CMAPI_CDMA2000_ActivateManual()	command the device to perform manual activation using the specified parameters	1.0
CMAPI_CDMA2000_ValidateSPC()	command the device to validate a Service Programming Code (SPC)	1.0
CMAPI_OMADM_StartSession()	start an OMA DM session to configure the values of various CDMA2000 network information as specified by the session type in its input parameter	1.0
CMAPI_OMADM_CancelSession()	cancel an ongoing OMA DM session	1.0
CMAPI_OMADM_GetSessionInfo()	return information about the currently active OMA DM session (or the most recent session if none is active)	1.0
CMAPI_OMADM_GetPendingNIA()	return 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	1.0
CMAPI_OMADM_SendSelection()	return the response from the device to a Network-Initiated Alert (NIA) that is commanding the device to establish a DM session	1.0
CMAPI_OMADM_GetFeatureSettings()	return information about the settings of OMA DM features, indicating for each one whether OMA DM can be currently used for the specified configuration operation	1.0
CMAPI_OMADM_SetProvisioningFeature()	enable and disable the OMA DM device service provisioning update feature	1.0
CMAPI_OMADM_SetPRLUpdateFeature()	enable and disable the OMA DM PRL update feature	1.0
CMAPI_OMADM_SetFirmwareUpdateFeature() (Optional)	enable and disable the OMA DM Firmware update feature	1.0
CMAPI_OMADM_ResetToFactoryDefaults()	reset the device to factory default	1.0
CMAPI_OMADM_InitiateOTASP()	activate the device using OTA activation	1.0
CMAPI_OMADM_SetPRL()	update PRL/PLMN by uploading a PRL file	1.0
CMAPI_MobileIP_SetState()	set the current Mobile IP state of the device	1.0
CMAPI_MobileIP_GetState()	retrieve the current Mobile IP state of the device	1.0
CMAPI_MobileIP_SetActiveProfile()	set the index of the Mobile IP profile that the device will use	1.0
CMAPI_MobileIP_GetActiveProfile()	retrieve the index of the Mobile IP profile that the device is currently using	1.0
CMAPI_MobileIP_SetProfile()	configure the contents of a Mobile IP profile on the device	1.0
CMAPI_MobileIP_GetProfile()	retrieve the contents of a Mobile IP profile on the device	1.0
CMAPI_MobileIP_SetParameters()	set various parameters that configure the behaviour of the device’s Mobile IP client	1.0
CMAPI_MobileIP_GetParameters()	retrieve the current values of the parameters that configure the behaviour of the device’s Mobile IP client	1.0
CMAPI_MobileIP_GetLastError()	retrieve the last Mobile IP error that occurred (refer to RFC3344 for a list of error codes)	1.0
DEVICE SERVICE APIs		
CMAPI_DevSrv_GetManufacturerName()	retrieve the name of the manufacturer of the device	1.0
CMAPI_DevSrv_GetManufacturerModel()	retrieve the product model ID of the device	1.0
CMAPI_DevSrv_GetDeviceName()	retrieve the commercial name of the device	1.0
CMAPI_DevSrv_GetHardwareVersion()	retrieve the hardware version of the device	1.0
CMAPI_DevSrv_GetProductType()	retrieve the product type of the device	1.0

CMAPI_DevSrv_GetIMSI()	retrieve the active IMSI(s) from SIM/R-UIM/NAA on UICC	1.0
CMAPI_DevSrv_GetMDN()	retrieve the MDN (only applicable to 3GPP2 systems)	1.0
CMAPI_DevSrv_GetIMEI()	retrieve the IMEI (only applicable to 3GPP systems)	1.0
CMAPI_DevSrv_GetESN()	retrieve the ESN (only applicable to 3GPP2 systems)	1.0
CMAPI_DevSrv_GetMEID()	retrieve the MEID (only applicable to 3GPP2 systems)	1.0
CMAPI_DevSrv_GetMSISDN()	retrieve the MSISDN from the active NAA in the SIM/UICC (only applicable to 3GPP systems)	1.0
CMAPI_DevSrv_GetDeviceStatus()	retrieve the device status	1.0
CMAPI_DevSrv_GetFirmwareVersion()	retrieve the firmware version of the device	1.0
CMAPI_DevSrv_GetRFSwitch()	retrieve the radio switch status (Radio On / Off)	1.0
CMAPI_DevSrv_SetRadioState()	set the radio power state of the device	1.0
CMAPI_DevSrv_SetRadioState_Async()	set the power state of a radio within a device	1.0
CMAPI_DevSrv_GetControlKeyStatus()	get the specified Mobile Equipment (device) de-personalization control key status	1.0
CMAPI_DevSrv_DeactivateControlKey()	deactivate the specified Mobile Equipment (device) de-personalization control key	1.0
CMAPI_DevSrv_UnblockControlKey() (Optional)	unblock the specified Mobile Equipment (device) de-personalization control key	1.0
CMAPI_DevSrv_DevAttributes()	provide to application information regarding device attributes (e.g. screen, keypad, camera, microphone, loudspeaker)	1.1
DEVICE EXTENDED SERVICE APIs		
CMAPI_ExtDevSrv_NFC()	provide to application information regarding NFC functionalities available in the device	1.1
CMAPI_ExtDevSrv_SE()	provide to application information regarding SE (Secure Element) functionalities and services available in the device	1.1
PINS/PUKS MANAGEMENT APIs		
CMAPI_DevSrv_GetNAAavailable()	get all the available NAAs and the corresponding Application labels	1.0
CMAPI_DevSrv_EnablePIN()	enable PIN protection	1.0
CMAPI_DevSrv_DisablePIN()	disable PIN protection	1.0
CMAPI_DevSrv_VerifyPIN()	verify a PIN	1.0
CMAPI_DevSrv_UnblockPIN()	unblock a PIN	1.0
CMAPI_DevSrv_ChangePIN()	change a PIN	1.0
UICC MANAGEMENT APIs		
CMAPI_UICC_GetTerminalProfile()	get the last TERMINAL PROFILE sent by the device to the SIM/R-UIM/UICC	1.0
CMAPI_UICC_SetTerminalProfile()	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	1.0
CMAPI_UICC_SendToolKitEnvelopeCommand()	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	1.0
CMAPI_UICC_SendTerminalResponse()	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()**	1.0
WLAN APIs		
CMAPI_WLAN_IsSupported()	determine if WLAN functionality is supported	1.0
CMAPI_WLAN_AddKnownNetwork()	add a network to the known network list	1.0
CMAPI_WLAN_UpdateKnownNetwork()	update an existing known network record	1.0
CMAPI_WLAN_DeleteKnownNetwork()	remove the entry from the known networks list at the specified index	1.0
CMAPI_WLAN_GetKnownNetwork()	retrieve the known network record information	1.0
CMAPI_WLAN_GetScanResults()	retrieve the list of available WLAN networks	1.0
CMAPI_WLAN_Scan_Async()	initiate a scan for WLAN networks	1.0
CMAPI_WLAN_Connect()	connect to a WLAN network	1.0
CMAPI_WLAN_ConnectKnownNetwork()	connect to a WLAN network in the known networks list	1.0
CMAPI_WLAN_Disconnect()	disconnect any connected WLAN network	1.0

CMAPI_WLAN_GetConnectionMode()	determine if connectivity is being actively sought by the enabler or if manual connection requests are required	1.0
CMAPI_WLAN_SetConnectionMode()	change the connectivity mode	1.0
CMAPI_WLAN_ResetDevice()	reset the device	1.0
CMAPI_WLAN_GetConnectedParameters()	retrieve values related to the associated network.	1.0
CMAPI_WLAN_SetConnectedParameters()	set various attributes of an existing connection	1.0
CMAPI_WLAN_CancelOperation()	cancel any pending operation like connect or scan	1.0
CMAPI_WLAN_ConnectWPS()	initiate a connection with the WPS button push method.	1.0
CMAPI_WLAN_ConnectPinWPS()	initiate a connection with the WPS pin method	1.0
CMAPI_WLAN_ConnectionState()	determine if WLAN is connected	1.0
CMAPI_WLAN_SearchNetwork_Async()	check the availability of a specific WLAN network	1.0
CMAPI_WLAN_EnableCapability()	enable or disable the WLAN feature in the device	1.1
CMAPI_WLAN_AuthenticationSupported()	determine if HS2.0 is supported by the device and what are the authentications methods supported	1.1
CMAPI_WLAN_ManageKnownNetwork()	add/delete or update a network to or in the known network list.	1.1
CMAPI_WLAN_Get_WSIDL()	retrieve the user preferred list and operator preferred list of WLAN specific identifier (WSID) from the SIM/R-UIM/NAA on UICC	1.1
CMAPI_WLAN_Get_HS2MOSubcription()	retrieve the elements related to HS2.0 subscriptions	1.1
CMAPI_WLAN_Get_ANDSFMOSubcription()	retrieve the elements related to ANDSF subscription	1.1
CMAPI_WLAN_GetANQP()	get the ANQP information including HS2.0 ANQP	1.1
CMAPI_WLAN_Get_WLANSettings()	retrieve the user and operator settings for WLAN.	1.1
CMAPI_WLAN_Set_WLANUserSettings()	set the user settings for WLAN.	1.1
STATISTICS APIs		
CMAPI_NetStatistic_GetConnectionStatistics()	obtain network traffic statistics info	1.0
CMAPI_NetStatistic_GetAllConnectionRecords()	retrieve all connection records.	1.1
CMAPI_NetStatistic_DeleteConnectionRecord()	delete a connection record.	1.1
INFORMATION STATUS APIs		
CMAPI_Information_GetPINStatus()	return the status of the PINs and PUKs of all active SIM/R-UIM/NAA on UICC for a dedicated device	1.0
CMAPI_Information_GetNetworkSelectionMode()	determine the network selection mode	1.0
CMAPI_Information_GetSignalStrength()	obtain the current signal strength value, the percentage of signal present and the signal quality	1.0
CMAPI_Information_GetCSNetworkRegistration()	determine if a circuit switched registration is present	1.0
CMAPI_Information_GetPSNetworkRegistration()	determine if a packet switched attachment is present	1.0
CMAPI_Information_GetAPN()	obtain the APN identifier	1.0
CMAPI_Information_GetIPAddress()	retrieve the current IP address assigned to the device and the type of the address assigned	1.0
CMAPI_Information_GetRoamingStatus()	retrieve the current roaming status	1.0
CMAPI_Information_GetDriverVersion()	retrieve the driver version	1.0
CMAPI_Information_GetRATType()	retrieve the radio access technology	1.0
CMAPI_Information_GetQoS()	retrieve the QoS parameters related to the network	1.0
CMAPI_Information_GetWLANConnection()	retrieve identifying data of the currently connected network.	1.0
CMAPI_Information_GetRadioState()	return the power state of a radio within a device	1.0
CMAPI_Information_GetICCID()	get the ICCID	1.0
CMAPI_Information_GetBatteryStatus()	retrieve the current status of the battery of device if applicable	1.1
CMAPI_Information_SetBatteryThreshold()	set thresholds for the battery status	1.1
CMAPI_Information_GetMobilityState()	retrieve the current mobility state of the device	1.1
CMAPI_Information_GetMobilitytoLocation()	evaluate if the device is moving compared to a specific location (for example, an Access Point). The result is reported in callback **CMAPI_Callback_GetMobilitytoLocation_Complete()**	1.1
SMS MANAGEMENT APIs		
CMAPI_SMS_Send()	send SMS	1.0

CMAPI_SMS_Get()	retrieve the message	1.0
CMAPI_SMS_Delete()	delete SMS	1.0
CMAPI_SMS_GetIDList()	get the list of SMS stored on local device or SIM or the terminal device like PC	1.0
CMAPI_SMS_Update()	update the status of the SMS	1.0
CMAPI_SMS_GetSMSCAddress()	get the address of SMSC	1.0
CMAPI_SMS_SetSMSCAddress()	set the address of SMSC	1.0
CMAPI_SMS_GetValidityPeriod()	get the validity period setting	1.0
CMAPI_SMS_SetValidityPeriod()	set the period of validity of a SMS	1.0
CMAPI_SMS_GetDeliveryReport()	get the delivery report setting	1.0
CMAPI_SMS_SetDeliveryReport()	set the delivery report “On” or “Off”	1.0
CMAPI_SMS_GetRecordCount()	retrieve the number of SMS segments	1.0
CMAPI_SMS_GetUnreadRecordCount()	retrieve the number of unread SMS records	1.0
CMAPI_SMS_Create()	create a draft SMS	1.1
USSD MANAGEMENT APIs		
CMAPI_USSD_Request()	build up a USSD request to the network	1.0
CMAPI_USSD_Release()	release the USSD session	1.0
GNSS APIs		
CMAPI_GNSS_SetState()	set the state of the GNSS functionality on the device	1.0
CMAPI_GNSS_GetState()	retrieve the state of the GNSS functionality on the device	1.0
CMAPI_GNSS_SetTrackingParameters()	set the values of parameters that control the operation of GNSS tracking on the device	1.0
CMAPI_GNSS_GetTrackingParameters()	retrieve the values of parameters that control the operation of GNSS tracking on the device	1.0
CMAPI_GNSS_SetAGPSConfig()	configure the Assisted GPS (AGPS) server IP address, port number and/or FQDN	1.0
CMAPI_GNSS_GetAGPSConfig()	retrieve the values of the Assisted GPS (AGPS) server IP address, port number and FQDN	1.0
CMAPI_GNSS_SetAutomaticTracking()	enable and disable automatic GNSS tracking on the device	1.0
CMAPI_GNSS_GetAutomaticTracking()	retrieve the state of automatic GNSS tracking on the device	1.0
CMAPI_GNSS_GetDevicePosition()	retrieve the current position of the device	1.0
CMAPI_GNSS_SetSystemTime()	set the value of the system time	1.0
DATA PUSH SERVICE MANAGEMENT APIs		
CMAPI_Push_Enable()	turn on PUSH option	1.0
CMAPI_Push_Disable()	turn off PUSH option	1.0
CMAPI_Push_GetRadioType()	get the current bearer type over which the PUSH session is established for an application	1.0
CONTACT MANAGEMENT APIs		
CMAPI_Contact_Create()	create a contact	1.1
CMAPI_Contact_Get()	retrieve the details of a contact	1.1
CMAPI_Contact_Delete()	delete a contact	1.1
CMAPI_Contact_GetContactList()	get the list of contacts stored on local device or SIM or the terminal device like PC	1.1
CMAPI_Contact_Update()	update an existing contact	1.1
CMAPI_Contact_Search()	search for a specific contact name in the list of contacts	1.1
P2P DIRECT MANAGEMENT APIs		
CMAPI_P2P_GetP2PInfo()	detect which P2P direct connection technology(ies) is/are supported if any	1.1
CMAPI_P2P_EnableDirectDiscovery()	activate the P2P Direct Discovery Feature in a P2P Direct enabled device	1.1
CMAPI_P2P_DisableDirectDiscovery()	deactivate the P2P Direct Discovery feature in a P2P Direct enabled device	1.1
CMAPI_P2P_EnableDirectConnection()	activate the P2P Direct Connection feature in a P2P Direct enabled device	1.1
CMAPI_P2P_DisableDirectConnection()	deactivate the P2P Direct Connection feature in a P2P Direct	1.1

	enabled device	
CMAPI_P2P_DiscoveryResolve()	resolve a ServiceRecord for metadata and/or connection info	1.1
CMAPI_P2P_DiscoveryMonitor()	request discovery of Remote Device(s) and the services offered	1.1
CMAPI_P2P_DiscoveryAnnounce()	announce its presence and P2P Direct services supported to Remote Device(s)	1.1
CMAPI_P2P_EstablishConnection()	request the Local Device to establish a connection to a Remote Device	1.1
CMAPI_P2P_RejectConnection()	reject an incoming connection request	1.1
CMAPI_P2P_AcceptConnection()	accept an incoming connection request from a Remote Device	1.1
CMAPI_P2P_CloseConnection()	request the Local Device to close an existing connection to a Remote Device	1.1
CMAPI_P2P_GetConnectionStatus()	retrieve the status of the P2P Direct connection	1.1
CMAPI_P2P_EnableRelay()	request the Local Device to act as a relay to share its data connection with Remote Device members of the group (i.e. enable concurrent operations)	1.1
CMAPI_P2P_DisableRelay()	request the Local Device to stop acting as a relay to share its data connection with Remote Device members of the group	1.1
CMAPI_P2P_CreateGroup()	create a new P2P Direct group with one or several Remote Device (s)	1.1
CMAPI_P2P_RemoveGroup()	remove a P2P group, previously created by the Local Device	1.1
CMAPI_P2P_EnableMembershipInSeveralGroups()	enable a Local Device to be a member of several groups simultaneously	1.1
CMAPI_P2P_DisableMembershipInSeveralGroups()	disable a Local Device to be a member of several groups simultaneously	1.1
CMAPI_P2P_RemoveDeviceFromGroup()	remove a Remote Device from an existing group the Local Device owns	1.1
CMAPI_P2P_AcceptInvitationToGroup()	accept an group join invitation on the receiver side	1.1
CMAPI_P2P_JoinGroup()	invite a Remote Device to join an existing group	1.1
CMAPI_P2P_RejectInvitationToGroup()	reject an invitation to join an existing group	1.1
CMAPI_P2P_RejectJoiningGroup()	reject a Remote Device from joining to an existing group	1.1
CMAPI_P2P_RequestToJoinGroup()	send a request for joining an existing group to the group owner	1.1
CMAPI_P2P_RestrictFromGroup()	instruct the Local Device to be restricted from an existing group owned by a Remote Device	1.1
CMAPI_P2P_GetGroupInfo()	retrieve from the Local Device which P2P Direct enabled device(s) are in an existing group to which the Local Device is a member of	1.1
CMAPI_P2P_AllowSimultaneousConnection()	allow the device to have a P2P connection simultaneously to a normal data connection using the same radio technology.	1.1
CMAPI_P2P_DisallowSimultaneousConnection()	disallow the device to have a P2P connection simultaneously to a normal data connection using the same radio technology.	1.1
WIRELESS ROUTER APIs		
CMAPI_Router_GetConfigurations()	read the configuration values of a router (ssid, users, security, etc) of all defined routers of a physical router device	1.1
CMAPI_Router_SetConfiguration()	write the configuration values of a router (ssid, users, security, etc)	1.1
CMAPI_Router_DeleteConfiguration()	delete a router and its configuration	1.1
CMAPI_Router_GetConnectedDevices()	retrieve a list of Connected Devices connected to a router	1.1
CMAPI_Router_GetPolicies()	retrieve a list of policies within a router	1.1
CMAPI_Router_SetPolicy()	add or update a policy to a router's policies	1.1
CMAPI_Router_DeletePolicy()	delete a Connected Device policy from a router's policies	1.1
CMAPI_Router_GetRestrictions()	retrieve a list of Connected Device restrictions within a route	1.1
CMAPI_Router_SetRestriction()	add or update a Connected Device restriction to a router	1.1
CMAPI_Router_DeleteRestriction()	remove a Connected Device restriction	1.1
CMAPI_Router_SetAdminPassword()	update a router administrator password	1.1
CMAPI_Router_VerifyAdminPassword()	verify a router administrator password and to report the number of failed access attempts	1.1
CMAPI_Router_ResetToDefaults()	return a router to factory default settings	1.1
IP Multimedia Services APIs		
CMAPI_IMS_GetISIMinfo()	retrieve if there is an ISIM in the UICC for a specific radio system (either 3GPP or 3GPP2) and provide the IMPU, IMPI & Home	1.1

	Domain Name (relevant for IMS context) related.	
CMAPI_IMS_GetIARIinfo()	retrieve the IARI information from the ISIM for a dedicated radio system (either 3GPP or 3GPP2) on the UICC.	1.1
M2M/IoT APIs		
CMAPI_IoT_IMSI_Attach()	request an IMSI attach or detach.	1.1
CMAPI_IoT_GPRS_Register()	request an GPRS attach or detach.	1.1
CMAPI_IoT_Set_PDPContext()	define a PDP context.	1.1
CMAPI_IoT_GetPDPContextList()	get the list of currently defined PDP Contexts.	1.1
CMAPI_IoT_GetPDPContextIPaddress()	retrieve the IP address of the PDP context concerned.	1.1
CMAPI_IoT_Activate_PDPContext()	activate or deactivate a PDP context.	1.1
CMAPI_IoT_SetNFM()	set up (enable or disable) the Network Friendly Mode of the modem if supported	1.1
CMAPI_IoT_GetNFM()	return the current state of the Network Friendly Mode of the modem (enabled or disabled)	1.1
CMAPI_IoT_SetBack-OffBaseInterval()	configure the Back-off Base Intervals of the modem (time between re-attempts of whatever action previously failed)	1.1
CMAPI_IoT_GetBack-OffTimer()	retrieve the time left of the back-off Timer.	1.1

Table 11: List CMAPI-1 Functions

C.2 CMAPI-2 Functions

CMAPI-2		
Function	**Description**	**Vers.**
REGISTRATION APIs		
CMAPI_Callback_Register()	register for the callbacks which are expected to be received	1.0
CMAPI_Callback_Unregister()	turn off all callbacks or just some	1.0
CALLBACK APIs		
CMAPI_Callback_DetectDevicesComplete()	communicate that a search and validation of the devices in the system is complete	1.0
CMAPI_Callback_DeviceChanged()	communicate whenever there is a change in a given device state in particular indicate that a device has become present or been removed	1.0
CMAPI_Callback_GetNetworkList_Async_Complete()	result of a previous call made to **CMAPI_NetConnectSrv_GetNetworkList_Async()**.	1.0
CMAPI_Callback_Connect_Async_Complete()	result of a previous call to **CMAPI_NetConnectSrv_Connect_Async()**	1.0
CMAPI_Callback_Disconnect_Async_Complete()	result of a previous call to **CMAPI_NetConnectSrv_Disconnect()**	1.0
CMAPI_Callback_CancelConnect_Async_Complete()	result of a previous call to **CMAPI_NetConnectSrv_CancelConnect_Async()**	1.0
CMAPI_Callback_SessionStateChange()	communicate the session state change	1.0
CMAPI_Callback_BearerStatusChange()	communicate a bearer status change	1.0
CMAPI_Callback_TrafficChannelDormancy()	communicate the changes in the traffic level	1.0
CMAPI_Callback_CDMA2000ActivationState()	communicate the changes in the CDMA 2000 Activation state	1.0
CMAPI_Callback_SearchWLANNetworkComplete()	result of a previous call to **CMAPI_WLAN_SearchNetwork_Async()**	1.0
CMAPI_Callback_RadioState()	communicate changes in the radio power state	1.0
CMAPI_Callback_SetRadioState_Async_Complete()	result of a previous call to **CMAPI_DevSrv_SetRadioState_Async()**	1.0
CMAPI_Callback_Roaming()	indicate changes in Roaming status	1.0
CMAPI_Callback_SignalStrength()	return the current signal strength value, the percentage of signal present and the signal quality	1.0
CMAPI_Callback_GNSS()	indicate a change in the GNSS state	1.0
CMAPI_Callback_SMS()	indicate that a new SMS message has been received and the number of segments in the mailbox	1.0
CMAPI_Callback_SMS_Message()	provide to application the new received message while not only a notice that a new message is received	1.0
CMAPI_Callback_ByteCount	indicate the current byte count	1.0
CMAPI_Callback_USSD()	communicate a USSD message	1.0

CMAPI_Callback_QoSChange()	communicate a change in QoS	1.0
CMAPI_Callback_RFInformationChange()	communicate a change related to RF	1.0
CMAPI_Callback_PINPUKStatus()	return the status of the PINs/PUKs for all active NAAs	1.0
CMAPI_Callback_ScanWLANComplete()	notify that a scan for WLAN networks has been completed. result of a previous call to **CMAPI_WLAN_Scan_Async()**	1.0
CMAPI_Callback_WLANNewAvailableNetwork()	notify that a new network has been discovered	1.0
CMAPI_Callback_WLANConnectionStatus()	receive WLAN connection Status	1.0
CMAPI_Callback_PUSHReceived()	notify an application when a new PUSH message has been received	1.0
CMAPI_Callback_OMADMStatus()	indicate any OMA-DM operation Progress or Status in-between	1.0
CMAPI_Callback_UICC_ToolKitProactiveCommand()	receive the ToolKit Proactive Commands sent by the SIM/R-UIM/UICC	1.0
CMAPI_Callback_UICC_DeviceTerminalProfile()	receive the TERMINAL PROFILE sent by the device to the SIM/R-UIM/UICC	1.0
CMAPI_Callback_VerifyPIN()	signal that a PIN should be collected from the user and supplied to the API through the **CMAPI_DevSrv_VerifyPIN()** method	1.0
CMAPI_Callback_PermittedBearersChange()	notify that a change occurred in the PermittedBearers for the device	1.0
CMAPI_Callback_NetConnectSrv_SecondaryPDPContext_Connect_Async_Complete()	result of a previous call to **CMAPI_NetConnectSrv_SecondaryPDPContext_Connect_Async()**	1.0
CMAPI_Callback_NetConnectSrv_SecondaryPDPContext_Disconnect_Async_Complete()	result of a previous call to **CMAPI_NetConnectSrv_SecondaryPDPContext_Disconnect_Async()**	1.0
CMAPI_Callback_NetConnectSrv_SecondaryPDPContext_CancelConnect_Async_Complete()	result of a previous call to **CMAPI_NetConnectSrv_SecondaryPDPContext_CancelConnect_Async()**	1.0
CMAPI_Callback_WLANSettingsChanged()	notify that a WLAN operator setting has been changed.	1.1
CMAPI_Callback_WLANNewMO()	notify that a new or an updated WLAN MO has been provided to the Terminal.	1.1
CMAPI_Callback_Incoming_Voice_Call()	provide to application information regarding a voice call state (incoming, established...)	1.1
CMAPI_Callback_SEServicesChange()	communicate changes regarding the availability of services	1.1
CMAPI_Callback_BatteryStatusChanged()	communicate whenever there is a change in the battery status	1.1
CMAPI_Callback_BatteryThresholdReached()	communicate whenever the battery level of the device is reaching a threshold set by the function **CMAPI_Information_SetBatteryThreshold()**	1.1
CMAPI_Callback_P2P_DiscoveryMatch()	alert the Local Device of a DeviceID/ServiceID/ServiceRecord discovery match as indicated in **CMAPI_P2P_Monitor()**	1.1
CMAPI_Callback_P2P_Connection()	result of call made to **CMAPI _P2P_EstablishConnection()**	1.1
CMAPI_Callback_P2P_GroupNotification()	result of a previous call to **CMAPI_P2P_JoinGroup()**	1.1
CMAPI_Callback_GetMobilitytoLocation_Complete()	result of a previous call to **CMAPI_Information_GetMobilitytoLocation()**	1.1

Table 12: List CMAPI-2 Functions

Appendix D. Typical scenario for use of OpenCMAPI in Mobile Broadband - Laptop context (Informative)

D.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

D.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.

D.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.

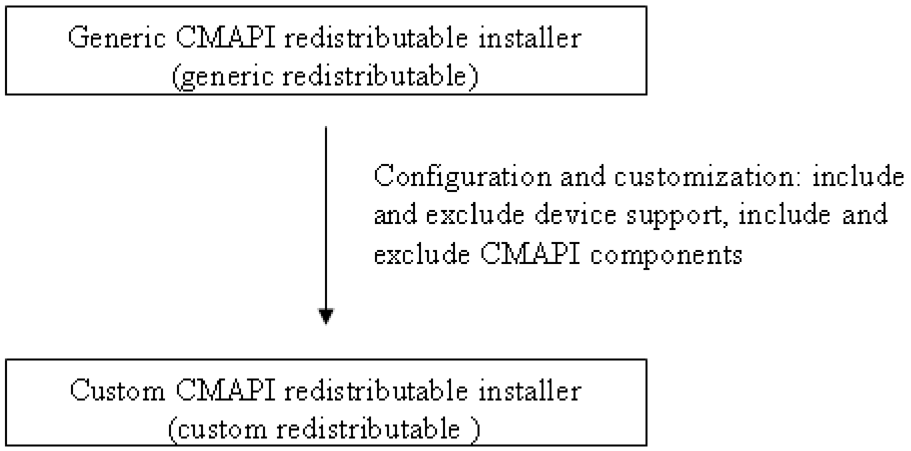
Generic CMAPI redistributable installer
(generic redistributable)
Configuration and customization: include and exclude device support, include and exclude CMAPI components
Custom CMAPI redistributable installer
(custom redistributable)

Figure 3: Configuration of OpenCMAPI redistributable installer

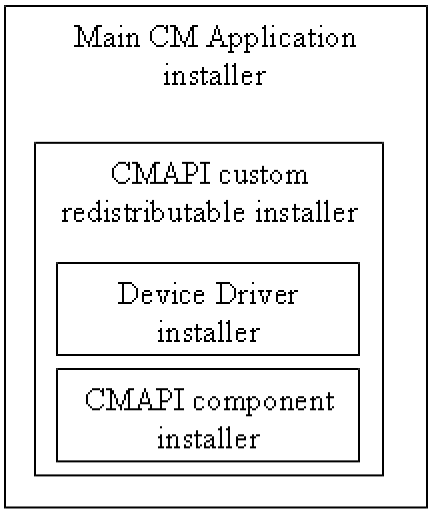
Main CM Application installer
CMAPI custom redistributable installer
Device Driver installer
CMAPI component installer

Figure 4: Example of CM Application installer

Appendix E. Consideration for implementation (Informative)

E.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.

E.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.

E.3 Implementation aspects

E.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.

E.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.

E.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.

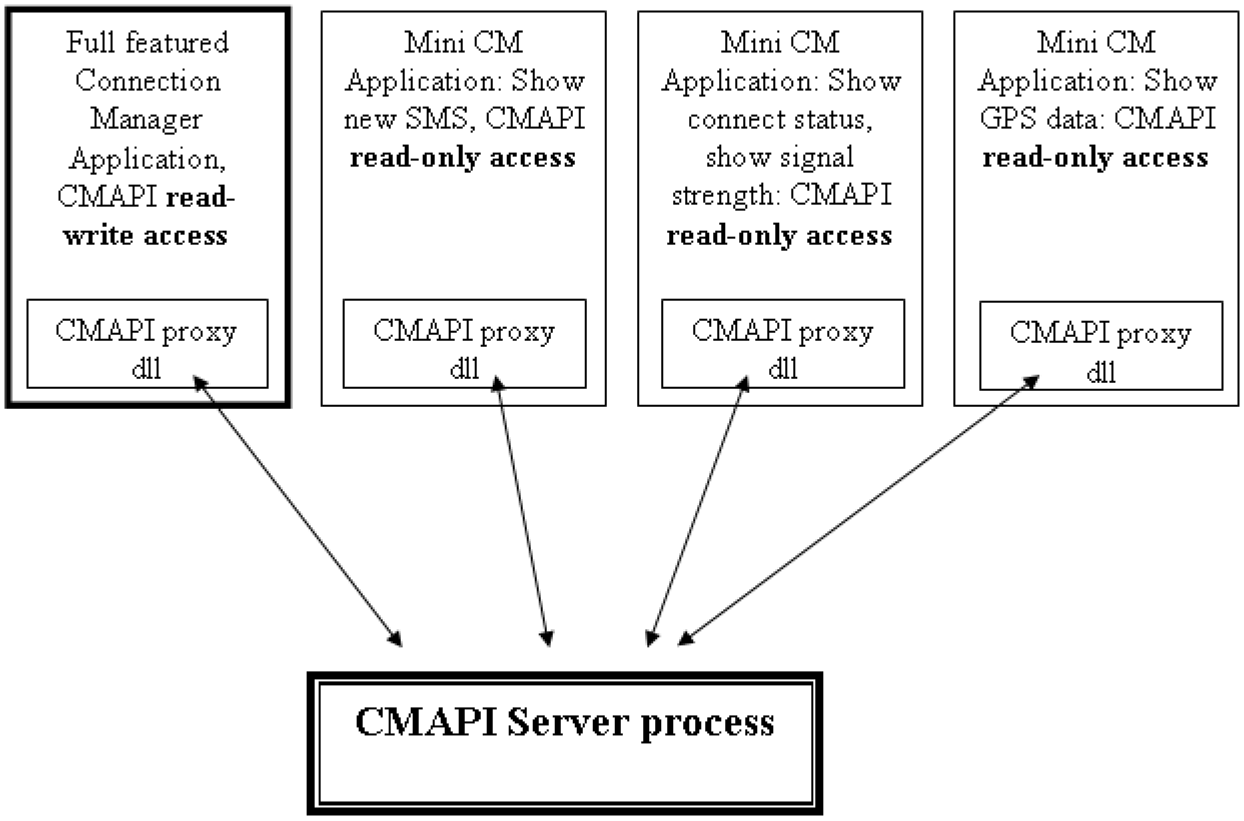
Full featured Connection Manager Application, CMAPI **read-write access**
CMAPI proxy dll
Mini CM Application: Show new SMS, CMAPI **read-only access**
CMAPI proxy dll
Mini CM Application: Show connect status, show signal strength: CMAPI **read-only access**
CMAPI proxy dll
Mini CM Application: Show GPS data: CMAPI **read-only access**
CMAPI proxy dll
CMAPI Server process

Figure 5: Open CM API as a server process

E.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.