International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

J.1033
(08/2020)

SERIES J: CABLE NETWORKS AND TRANSMISSION OF TELEVISION, SOUND PROGRAMME AND OTHER MULTIMEDIA SIGNALS

Conditional access and protection – Downloadable conditional access system for bidirectional networks

Downloadable conditional access system for bidirectional networks – The terminal

Recommendation ITU-T J.1033

Recommendation ITU-T J.1033

Downloadable conditional access system for bidirectional networks – The terminal

Summary

Recommendation ITU-T J.1033 specifies a terminal for the two-way downloadable conditional access system (DCAS) for bidirectional networks. A two-way DCAS protects broadcast content/services and controls consumer entitlements in the same way as what traditional conditional access (CA) systems do, and enables a two-way terminal device, such as a set-top-box (STB), to adapt to a new CA system by downloading and installing a new CA system's client software without changing hardware. In particular, a two-way DCAS can work in bidirectional cable television networks and other bidirectional networks such as broadband cable networks.

History

Edition	Recommendation	Approval	Study Group	Unique ID[*]
1.0	ITU-T J.1033	2020-08-13	9	11.1002/1000/14356

Keywords

Bidirectional network, CA, DCAS, downloadable.

[*] To access the Recommendation, type the URL http://handle.itu.int/ in the address field of your web browser, followed by the Recommendation's unique ID. For example, http://handle.itu.int/11.1002/1000/11830-en.

Table of Contents

Introduction

This Recommendation is part 3 of a multi-part deliverable covering the terminal for the two-way DCAS specification, as identified below:

Part 1 [ITU-T J.1031]:	"Requirements";
Part 2 [ITU-T J.1032]:	"System Architecture";
Part 3 (ITU-T J.1033):	"**The Terminal**".

Recommendation ITU-T J.1033

Downloadable conditional access system for bidirectional networks – The terminal

1 Scope

The object of this Recommendation is to specify the terminal of the two-way DCAS for the bidirectional network, including the terminal security chipset, the two-way DCAS client software and the related application programming interfaces (APIs). This Recommendation is one in a series of Recommendations, specifying the whole two-way DCAS for the bidirectional network. The other parts of two-way DCAS Recommendations include the requirement for the two-way DCAS as defined in [ITU-T J.1031], and the specification of system architecture and related security mechanism for the two-way DCAS as defined in [ITU-T J.1032].

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[ITU-T J.1031]	Recommendation ITU-T J.1031 (2020), *Downloadable conditional access system for bidirectional networks – Requirements.*
[ITU-T J.1032]	Recommendation ITU-T J.1032 (2020), *Downloadable conditional access system for bidirectional networks – System architecture.*
[ETSI ETR 289]	ETSI ETR 289 (1996), *Digital Video Broadcasting (DVB); Support for use of scrambling and Conditional Access (CA) within digital broadcasting systems.*
[NIST FIPS 197]	NIST U.S. FIPS PUB 197 (FIPS 197) (2001), *Advanced Encryption Standard (AES).*
[NIST 3DES]	National Institute of Standards and Technology (2017), *Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher (Revision 2), NIST Special Publication 800-67, November*.
[ISO/IEC 10118-3]	ISO/IEC 10118-3:2018, *Information technology – Security techniques – Hash-functions – Part 3: Dedicated hash-functions*.
[ISO/IEC 13818-1]	ISO/IEC 13818-1:2019, *Information technology – Generic coding of moving pictures and associated audio information – Part 1: Systems*.
[ISO/IEC 14888-3]	ISO/IEC 14888-3:2018, *Information technology – Security techniques – Digital signatures with appendix – Part 3: Discrete logarithm based mechanisms*.

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

3.1.1 descrambling [b-ITU-T J.93]: The processes of reversing the scrambling functions (see "scrambling") to yield usable pictures, sound and data services.

3.1.2 entitlement control messages (ECMs) [b-ITU-T J.290]: An ECM is an encrypted message that contains access criteria to various service tiers and a control word (CW).

3.1.3 entitlement management messages (EMMs) [b-ITU-T J.290]: The EMM contains the actual authorization data and shall be sent in a secure method to each CPE device.

3.1.4 scrambling [b-ITU-T J.93]: The process of using an encryption function to render television and data signals unusable to unauthorized parties.

3.1.5 bootloader [b-ITU-T J.1026]: The program for initiating hardware and loading software after a receiver boots up.

3.1.6 downloadable conditional access system (DCAS) [b-ITU-T J.1026]: A conditional access (CA) system that supports all the features of a legacy conditional access and provides CA-neutral mechanism to securely download CA client image and switch CA terminals without changing hardware through either a broadcasting or two-way network.

3.1.7 key ladder (KLAD) [b-ITU-T J.1026]: A structured multi-level key mechanism that ensures secure transport of control word.

3.1.8 root key [b-ITU-T J.1026]: The key used for the first level of a key ladder.

3.1.9 secure data management platform (SDMP) [b-ITU-T J.1026]: A platform that generates and manages some basic and root information, such as keys and IDs used in DCAS, including information to DCAS headend and to terminal security chipset.

3.1.10 security chipset key de-obfuscation [b-ITU-T J.1026]: Algorithm used to de-obfuscate encrypted security chipset key.

3.1.11 terminal security chipset [b-ITU-T J.1026]: A stream processing chipset with security functions such as secure key deriving and key ladder processing, etc.

3.1.12 terminal software platform [b-ITU-T J.1026]: A software platform running on a terminal, integrated with various hardware drivers, having various terminal application APIs, capable of downloading and running terminal applications according to specified security requirements, and providing a secure execution environment for terminal application.

3.1.13 transport stream filtering [b-ITU-T J.1026]: By using certain filtering mechanism to extract data matching filter rules from transport stream.

3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

3.2.1 two-way DCAS: A downloadable conditional access system (DCAS) operated especially in a two-way network.

3.2.2 two-way DCAS App: A two-way DCAS application running on the terminal software platform. After a terminal device is deployed in the field, this application can be upgraded or replaced through online downloading.

3.2.3 two-way DCAS trusted App: A two-way DCAS trusted application running in the trusted execution environment of a terminal device. After a terminal device is deployed in the field, this application can be upgraded or replaced through online downloading.

3.2.4 two-way DCAS manager: The terminal software platform's software component responsible for registering two-way DCAS client software, supporting information exchange between the two-way DCAS App and the two-way DCAS trusted App, as well as receiving and forwarding two-way DCAS entitlement control and management messages.

3.2.5 two-way DCAS client software: A terminal application composed of a two-way DCAS App and a two-way DCAS trusted App through the joint work with the support of the DCAS manager embedded in the terminal software platform.

3.2.6 two-way DCAS client software data: Data to be saved or updated, which include conditional access (CA) authorization information, CA private data, etc., when the two-way DCAS client software runs.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

API	Application Programming Interface
BOSS	Business Operations Support System
CA	Conditional Access
CAT	Conditional Access Table
CAS	Conditional Access System
CBC	Cipher Blocker Chaining
CCI	Copy Control Information
ChipID	Chipset Identification
CPU	Central Processing Unit
CSA	Common Scrambling Algorithm
CW	Control Word
CRC	Cyclic Redundancy Check
DCAS	Downloadable Conditional Access System
DVB	Digital Video Broadcasting
ECB	Electronic Code Book
ECM	Entitlement Control Message
EMM	Entitlement Management Message
EPG	Electronic Programme Guide
ESCK	Encrypted Security Chipset Key
GP	Global Platform
IP	Internet Protocol
IPTV	TV using the Internet Protocol (IP)
IPC	Inter-Process Communication

JS	JavaScript
KLAD	Key Ladder
OSD	On-Screen Display
OTP	One Time Programmable
PID	Packet Identification
PIN	Personal Identification Number
PMT	Program Map Table
RAM	Random Access Memory
SCK	Security chipset Key
SCKv	Security chipset Key vendor
SDMP	Secure Data Management Platform
Seedv	Seed vendor
SIM	Security Implementation Mechanism
SMK	Secret Mask Key
SoC	System on Chip
TA	Trusted Application
TDES	Triple-DES
TEE	Trusted Execution Environment
TVOS	Smart TV Operating System
UDP	User Datagram Protocol
UI	User Interface
UUID	Universally Unique Identifier
Vendor_SysID	Vendor System Identification

5 Conventions

In this Recommendation:

The keywords **"is required to"** indicate a requirement which must be strictly followed and from which no deviation is permitted if conformance to this document is to be claimed.

The keywords **"is recommended"** indicate a requirement which is recommended but which is not absolutely required. Thus this requirement need not be present to claim conformance.

The keywords **"is prohibited from"** indicate a requirement which must be strictly followed and from which no deviation is permitted if conformance to this document is to be claimed.

The keywords **"can optionally"** indicate an optional requirement which is permissible, without implying any sense of being recommended. This term is not intended to imply that the vendor's implementation must provide the option and the feature can be optionally enabled by the network operator/service provider. Rather, it means the vendor may optionally provide the feature and still claim conformance with the specification.

In the body of this Recommendation and its annexes, the words *shall*, *shall not*, *should*, and *may* sometimes appear, in which case they are to be interpreted, respectively, as *is required to*, *is* *prohibited from*, *is recommended*, and *can optionally*. The appearance of such phrases or keywords in an appendix or in material explicitly marked as *informative* are to be interpreted as having no normative intent.

6 Overview of the two-way DCAS terminal

The two-way DCAS terminal includes the terminal security chipset, the DCAS client software and the terminal software platform. The two-way DCAS terminal is an essential part of the two-way DCAS, whose architecture can be referred to Figure 1 in part 2 [ITU-T J.1032].

The two-way DCAS terminal validates the user's entitlement and descrambles protected services to implement conditional access of services. The terminal software platform can securely download, update and replace the two-way DCAS client software.

This Recommendation mainly focuses on specifying the terminal security chipset, two-way DCAS APIs embedded in the two-way DCAS terminal software platform, and two-way DCAS manager through the definition of the two-way DCAS APIs.

7 Architecture of the two-way DCAS terminal

The architecture of the two-way DCAS terminal is shown in the figure 1.

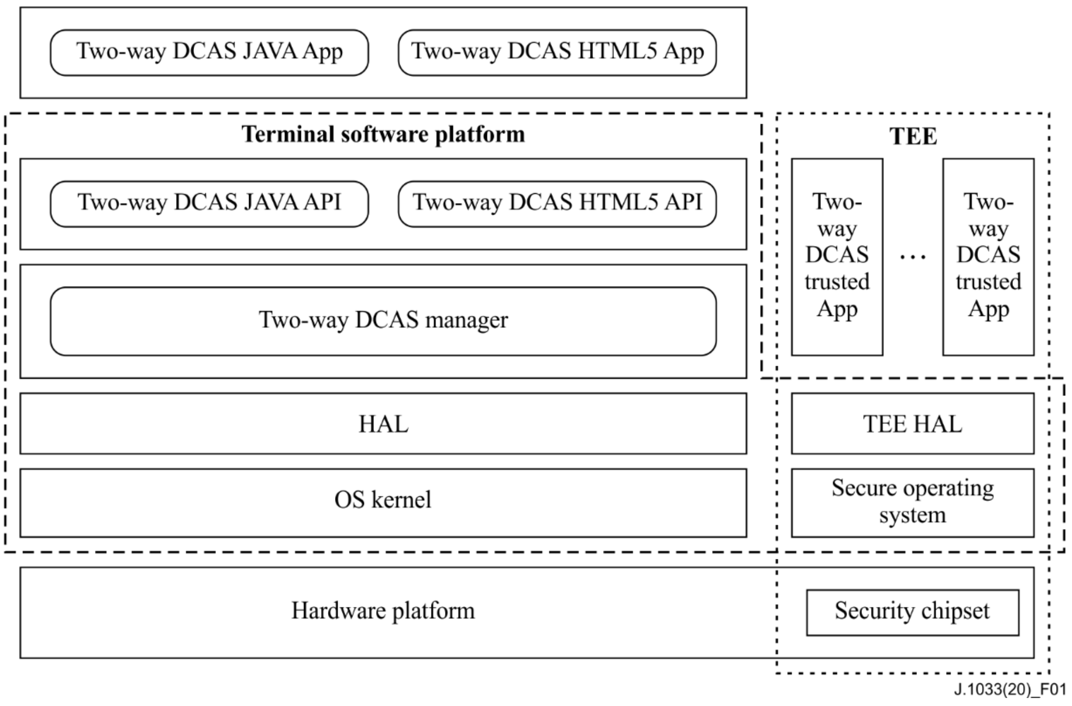

Figure 1 –Architecture of the two-way DCAS terminal

The terminal security chipset provides key ladder module and root key derivation module, to ensure the secure transport of the terminal data and the independence of the CA system. Clause 9 gives more details.

The two-way DCAS APIs embedded in the two-way terminal software platform support the joint work of two-way DCAS App and the two-way DCAS TApp with the assistance of the two-way DCAS manager embedded in the two-way terminal software platform.

The two-way DCAS manager uses its functions such as registration, cancellation and paring to manage the two-way DCAS App and the respective TApp.

The terminal software platform shall support trusted execution environment (TEE). It can either be a smart TV operating system (TVOS) or a middleware based on operating systems such as Linux and secureOS.

Two-way DCAS APIs supports DCAS manager to manage the two-way DCAS client software, and supports two-way DCAS client software to process the CA data such as ECM/EMM and data exchange with other applications such as electronic programme guide (EPG).

The two-way DCAS client software implements the functions of analysing and processing data such as ECM and EMM, and the processing of the secure information. The processing of the ECM and EMM shall be implemented within the TEE by DCAS TApp.

Functions of the two-way DCAS client software are achieved through the joint work of the two-way DCAS App and the respective two-way DCAS trusted App with support of the two-way DCAS manager and related APIs.

The two-way DCAS client software can be downloaded to terminal software platform, and runs in parallel with other applications on the same terminal software platform.

8 Two-way DCAS APIs

Two-way DCAS APIs are used to perform data exchange between two-way DCAS client software and terminal software platform.

There are two kinds of two-way DCAS APIs. One kind of API is the general API such as Java API and JavaScript API. The other kind of API is the two-way DCAS API for the two-way DCAS TApp. The detailed specification of the two-way DCAS APIs is given in Annex B.

The general APIs include:

a) Filtering APIs. Two-way DCAS client software invokes the filtering APIs to receive ECM, EMM and CAT.

b) Two-way DCAS management APIs. Two-way DCAS client software uses DCAS management APIs to register itself to the terminal software platform and receive descrambling requests from DCAS manager in the terminal software platform.

The two-way DCAS APIs for the two-way DCAS TApp include:

a) Key ladder APIs. Two-way DCAS client software invokes the key ladder APIs to load encrypted keys to the terminal security chipset to descramble transport streams.

b) Other global platform (GP) extension APIs. Two-way DCAS client software uses this APIs for encryption/decryption, signature verification, memory management, debug printing, etc.

9 Terminal security chipset

9.1 Terminal security chipset workflow

Figure 2 shows the functional diagram of the terminal security chipset also defined in [b-GY/T 255-2012], which includes modules of OTP, root key derivation, key ladder, descrambling and decoding.

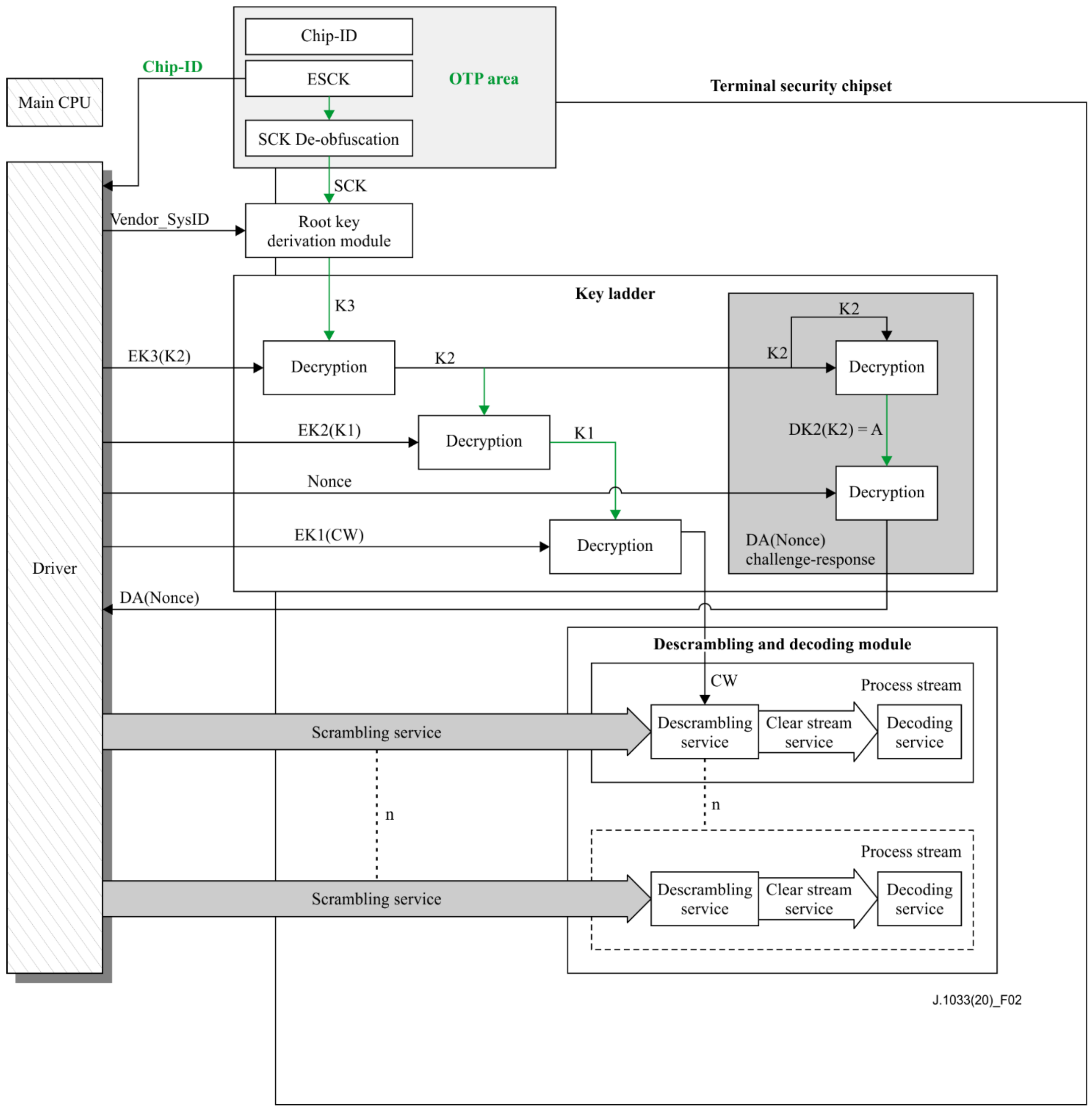

Figure 2 – Functional diagram of the terminal security chipset

The terminal security chipset uses root key derivation module to generate root key K3, and uses key ladder module to ensure the secure transport of the control word and other secret keys as well as the validity of the terminal security chipset.

The terminal security chipset's functionalities shall not be implemented by the primary central processing unit (CPU).

The workflow of the terminal security chipset is described as follows:

After a terminal security chipset is powered on, the OTP module of the terminal security chipset uses built-in encrypted security chipset key (ESCK) and security chipset key (SCK) de-obfuscation to generate SCK, which is fed to the root key derivation module to generate root key K3.

The key ladder module receives the root key K3 and uses it for decryption of keys and challenge-response. The key ladder module contains functions such as: decrypting keys level by level with the input encrypted keys; processing challenge information (Nonce) and generating response (DA (Nonce)).

Finally, the decrypted control word (CW) is sent to the descrambling and decoding module for descrambling and decoding services. The challenge-response is a function in the two-way system for the headend to certificate the system on chip (SoC) chip.

9.2 The root key derivation module

The root key derivation module consists of a group of hardware logical modules. It uses an embedded derivation mechanism together with input parameters for the derivation of the root key. All CA vendors use this derivation mechanism to generate different root keys. This method can circumvent the vulnerability of using a singular root key.

The functions of the root key derivation module include SCK preliminary manipulation function, vendor separation function and final root key derivation function. The root key derivation module can derive a specific root key for a two-way DCAS vendor according to the input SCK and Vendor_SysID. Figure 3 shows the functional diagram of the root derivation module also defined in [b-GY/T 255-2012] and [b-ETSI TS 103 162].

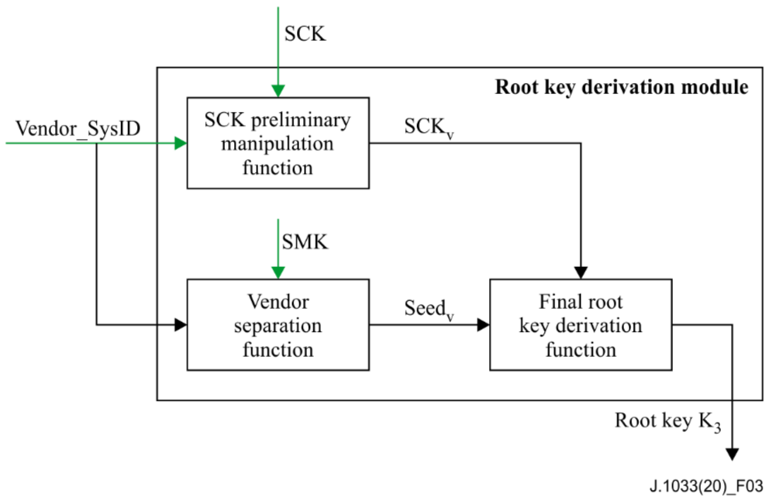
SCK
Root key derivation module
Vendor_SysID
SCK preliminary manipulation function
SCK$_v$
SMK
Vendor separation function
Seed$_v$
Final root key derivation function
Root key K$_3$
J.1033(20)_F03

Figure 3 – The functional diagram of the root derivation module

The preliminary SCK manipulation function generates security chipset key vendor (SCKv) based on SCK and input Vendor_SysID. The function must use a cipher algorithm with certain level of security strength which ensures the SCK cannot be retrieved when Vendor_SysID and SCKv are known. Additionally, the function shall be provided by the terminal security chipset vendor and certified by SDMP.

The vendor separation function generates Seedv by using Vendor_SysID and secret mask key (SMK) as input. The function must use a cipher algorithm with certain level of security strength which ensures the SMK cannot be retrieved when Vendor_SysID and Seedv are known. Additionally, the function shall be provided by the terminal security chipset vendor and certified by SDMP.

The final root key derivation function derives root key K3 based on the input of SCKv and Seedv. In the execution process, this function shall use a two-way function to ensure the other input parameter cannot be retrieved when K3 and any other input parameter are known. For example, Seedv cannot be retrieved when K3 and SCKv are known. This function shall be provided by the terminal security chipset vendor and certified by the SDMP.

– Vendor_SysID: 2 bytes, used for identifying CA system and assigned by SDMP.
– SCKv: 16 bytes
– SMK: 16 bytes, gate-level data provided by chipset vendor
– Seedv: 16 bytes
– Terminal security chipset can support various final root key derivation functions at the same time.

9.3 The key ladder module

9.3.1 3-level key mechanism

Figure 4 shows the functional diagram of the key ladder module also defined in [b-GY/T 255-2012].

The terminal security chipset specified by this Recommendation shall support the 3-level key ladder mechanism. The security requirements and technical details for the key ladder mechanism with more levels are out of the scope of this Recommendation.

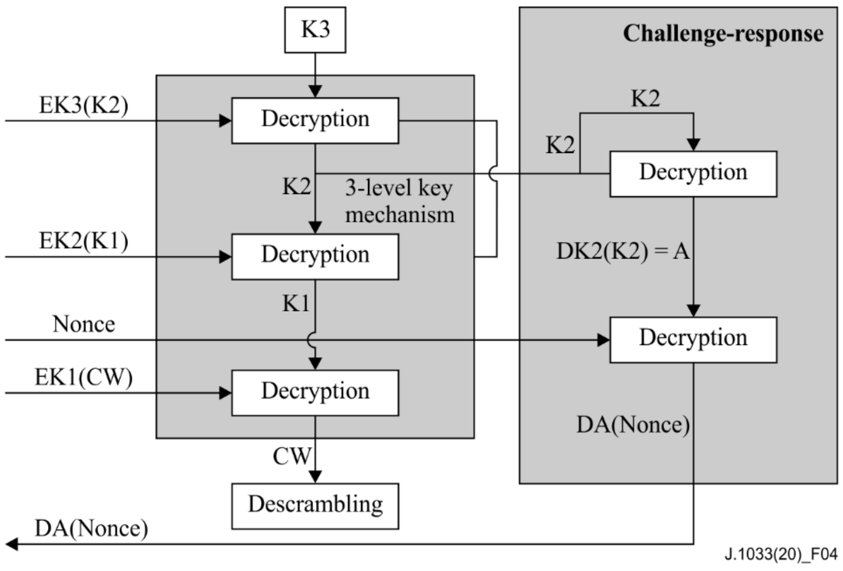

Figure 4 – The functional diagram of the key ladder module

The 3-level key ladder mechanism ensures secure transport of CW within the terminal.

The 3-level key ladder mechanism uses root key K3 obtained from root key derivation module, to decrypt EK3 (K2), EK2 (K1), EK1 (CW) to obtain the CW required by descrambling; at the same time, K2 works with challenge information (Nonce) to generate response(DA (Nonce)).

The terminal security chipset shall decrypt scrambled services with the following procedure:

a) Shall receive encrypted EK3(K2), use K3 to decrypt it and generate K2;
b) Shall receive EK2(K1), use K2 to decrypt it and generate K1;
c) Shall receive EK1(CW), use K1 to decrypt it and generate CW;
d) CW is used for decrypt scrambled services.

EK3(K2) represents key K2 encrypted with key K3.

EK2(K1) represents key K1 encrypted with key K2.

EK1(CW) represents CW encrypted with key K1.

K3 is derived root key, 16 bytes.

K2 is the key used to decrypt K1, 16 bytes.

K1 is the key used to decrypt CW, 16 bytes.

CW is key used to descramble services, 8 or 16 bytes

The key ladder mechanism may support SM4 algorithm defined in [b-GB/T 32907] with 128-bits keys and data blocks in electronic code book (ECB) mode.

9.3.2 Challenge-response mechanism

The terminal security chipset shall support challenge-response mechanism, which can be used in some security functions.

The challenge-response mechanism shall comply with the following procedure:

a) The terminal security chipset shall receive Vendor_SysID, EK3(K2) and Nonce by using the driver API.

b) The terminal security chipset shall use derived K3 to decrypt EK3(K2) to get K2.

c) The terminal security chipset shall decrypt K2 using K2 itself to generate DK2(K2), denoted as A.

d) The terminal security chipset shall decrypt Nonce using A, to generate DA(Nonce).

e) The terminal security chipset returns DA(Nonce) by using the driver API.

DK2(K2) represents the process of decrypting K2 using K2.

A denotes the result of DK2(K2), 16 bytes.

Nonce denotes challenge data, 16 bytes.

DA(Nonce) denotes the result from decrypting Nonce with A as the key.

9.3.3 The OTP area

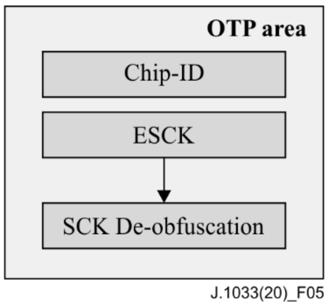

Figure 5 – OTP area functional diagram

Figure 5 shows the functional diagram of the OTP area also defined in [b-GY/T 255-2012], which is used to store information such as chipset identification (ChipID), ESCK and SCK de-obfuscation. The logical circuit of SCK de-obfuscation reads ESCK from the OTP area, de-obfuscates ESCK to SCK and provides SCK to root key derivation module.

SCK is the security information required for deriving root key. SCK should not be stored in the OTP area as plain text.

a) SCK is terminal security chipset key, which is unique per chipset and is generated by SDMP, 16 bytes.

b) ESCK is the encrypted SCK provided by SDMP. It is stored in OTP area with the same length as SCK.

c) ChipID is an 8-byte public identifier of terminal security chipset. It includes information such as chipset vendor, type, as well as a 4-byte unique global identifier assigned by secure data management platform. The ChipID format is described in Table 1.

Table 1 – ChipID data format

Data name	Length (Bit)	Data type
Chip Vendor ID	8	Uimsbf
Chip Type	12	Uimsbf
Reserved	12	Uimsbf
Chipset SN	32	Uimsbf

Chip Vendor ID: unique ID for chipset vendor, 8 bits.

Chip Type: ID for a chipset model manufactured by a chipset vendor. Assigned by SDMP, 12 bits.

Reserved: 12 bits.

Chipset SN, a 32-bit serial number of a chipset manufactured by a chipset vendor. It is globally unique for any chipset regardless of whether the chipset vendor or type is the same.

9.4 Security implementation mechanism (SIM)

9.4.1 SIM of the terminal security chipset

9.4.1.1 OTP area

The OTP area of terminal security chipset is used to store information such as ChipID, ESCK and BL_KEY0, etc. It shall comply with the following requirements:

a) The content written in OTP area cannot be changed;

b) SCK is necessary security information required for root key derivation, clear SCK should not be directly stored in the OTP area;

c) The security information in the OTP area shall be able to be tested and verified by the terminal security chipset to see if it has been tampered. If the security information is tampered, two-way DCAS function modules shall stop working immediately.

9.4.1.2 The SCK De-obfuscation function

The SCK de-obfuscation function of terminal security chipset shall comply with the following requirements:

a) It shall be a cryptographically secure enough two-way function.

b) It shall be implemented by internal hardware logical circuits; external software cannot intercept SCK or ESCK.

c) The SCK de-obfuscation function shall only be used by the root key derivation module.

9.4.1.3 Root key derivation module

The root key derivation module of terminal security chipset consists of sub modules such as SCK preliminary processing function, vendor separation function and final root key derivation function. The root key derivation module shall comply with the following requirements:

a) The root key derivation module shall be implemented by using hardware logical circuits or independent secure operation unit, working independently with dedicated calculation and storage resources. Any other units cannot interfere with the logic, execution, and result of the derivation module;

b) Any intermediate result from running of root key derivation module shall not be output or read to external modules.

c) SCK preliminary processing function shall use a two-way function with certain level of security strength, such as at least AES128 or SM4, to ensure the SCK cannot be de-obfuscated in case Vendor_SysID and SCKv are known. AES is defined in [NIST FIPS 197].

d) The vendor separation function shall use a two-way function with certain level of security strength to ensure SMK cannot be de-obfuscated in case Vendor_SysID and Seedv are known.

e) The final root key derivation function shall use a two-way function with a certain level of security strength so that when root key K3 and any input parameter are known, another input parameter cannot be retrieved. For example, Seedv cannot be retrieved when K3 and SCKv are known.

f) SMK shall not be read or tampered by external modules, and its length shall be at least 128 bits;

g) Terminal security chipset can support a variety of final root key derivation functions at the same time.

9.4.1.4 Key Ladder module

The key ladder module of the terminal security chipset consists of multi-level key mechanism and challenge-response mechanism. The key ladder module shall comply with the following requirements:

a) It must be implemented using hardware logical circuits or independent secure operation unit, and it must work independently with dedicated calculation and storage resources.

b) The logic, execution and results of the key ladder module cannot be interfered with or used by any other modules except the descrambling module can get CW from key ladder module.

c) Any intermediate result from running the key ladder shall not be output to or read out by any external module.

d) Any key in key ladder cannot be exposed in the challenge-response process.

Annex A

Security mechanism of two-way DCAS client software downloading and bootloading

(This annex forms an integral part of this Recommendation.)

A.1 Basic principles of chain of trust

The security mechanism of DCAS client software bootloading is based on a bottom-to-top chain of trust.

This chain of trust is established by using the digital signature technique. From bottom to top, it includes: terminal security chipset, bootloader, terminal software platform and DCAS client software.

The security mechanism of DCAS client software bootloading requires every link in the chain of trust must perform signature verification in a bottom-to-top order. Only if the signature verification at current link passes, can the security verification of next link be started. Only if the signature verification at all links passes, can DCAS client software be launched.

In the entire chain of trust:

a) The OTP area of terminal security chipset shall be preset with a verification key for bootloader.

b) Every link in the chain of trust shall be preset with a verification key for verifying software of the next link.

c) Software of every link of the chain of trust shall be digitally signed by a private key corresponding to the verification key.

d) Software of every link shall have its digital signature;

e) Software of the current link shall complete the security verification of the software of the next link first, before the software of the next link can be launched.

   The downloading, loading and running of DCAS client software shall comply with the mechanism of chain of trust described above at any time.

A.2 Bootup signature verification

The CPU shall execute certain secure codes to check the bootloader in the Flash that is already signed. Bootloader needs to verify the signature of the terminal software platform, and terminal software platform needs to verify signed DCAS client software.

See Figure A.1 for the verification process of the bootloader.

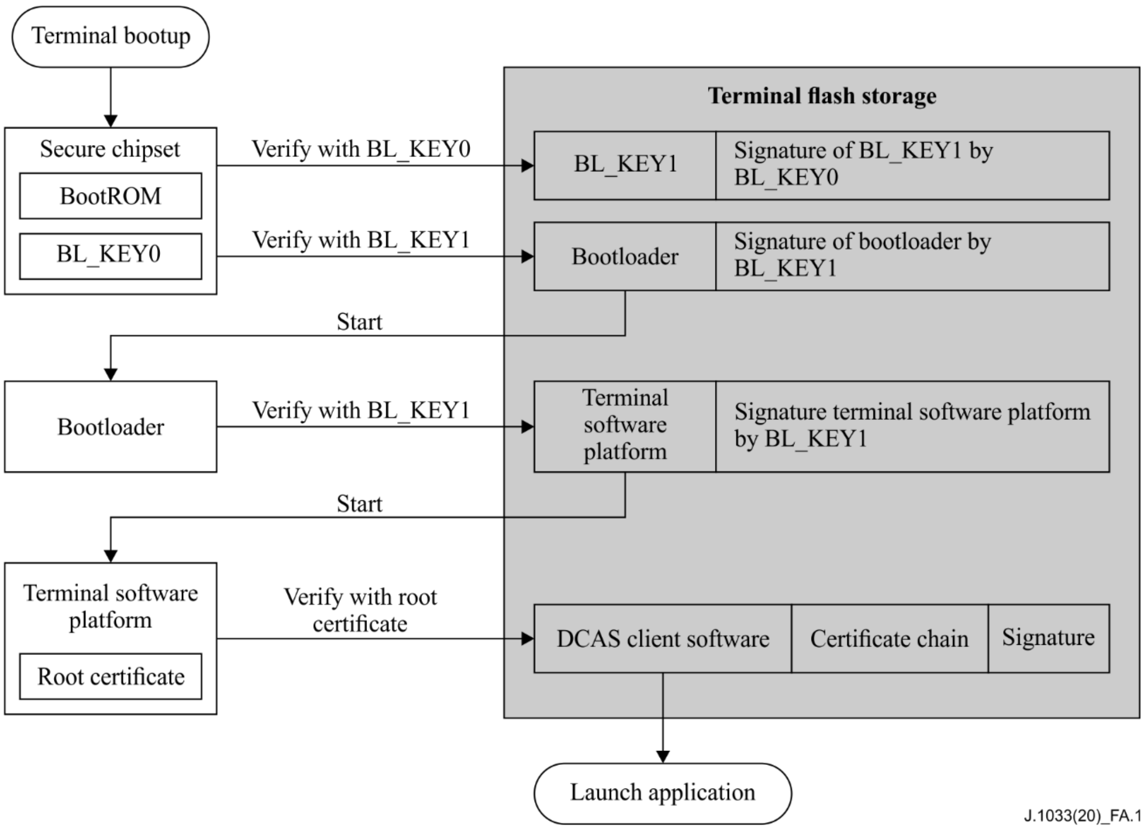

Figure A.1 – Verification of Bootloader

A public key assigned by SDMP, denoted as BL_KEY0, shall be embedded in the secure area of the terminal security chipset during SoC production. SDMP shall under no circumstances leak BL_KEY0's private key. The BL_KEY0 in Figure A.1 is the public key.

During bootup, the code of the BootRom in the secure area of terminal security chipset executes to read an additional public key (denoted by BL_KEY1) and signature of BL_KEY1 from terminal flash storage, and use BL_KEY0 to verify BL_KEY1's signature. The BL_KEY1 in Figure A.1 is the public key.

After BL_KEY1 is successfully verified, BootRom shall verify bootloader using BL_KEY1.

Bootloader shall use BL_KEY1 to verify the signature of terminal software platform. The terminal software platform shall use its embedded root certificate to verify the DCAS client software.

The algorithms used by all the signatures mentioned above include SM3 hash and SM2 public key cryptographic algorithm. SM3 algorithm is defined in [ISO/IEC 10118-3] and SM2 algorithm is defined in [ISO/IEC 14888-3].

A.3 Downloading and replacing the DCAS client software

The downloading and replacing of the DCAS client software, including DCAS App and DCAS trusted App, are implemented by the loader function of bootloader or the application manager in terminal software platform. The bootloader is used for the downloading and replacing of the whole terminal software image including terminal software platform and applications software, and the application manager is only used for the download and replacement of the applications software. The download and replacement procedure is as follows:

a) The headend system sends information of starting DCAS download and replacement to the terminal;

b) The bootloader or application manager downloads the image or DCAS client software;

c) The bootloader or application manager performs signature verification over the downloaded image or DCAS client software;

e) The bootloader or application manager replaces the image or DCAS client software with the verified new one;

e) The terminal reboots;

f) The new DCAS client software shall be able to work with SoC. Thus, the download and replacement of the DCAS client software are complete.

A.4 Key management

See Table A.1 for a description of keys for bootloader.

Table A.1 – Description of keys for bootloader

Key name	Key owner	Signed by	Used to sign	Note
BL_KEY0	SDMP	N/A	BL_KEY1	Public key is embedded to chipset
BL_KEY1	Operator	BL_KEY0	Bootloader Terminal software platform.	

For BL_KEY0:

SDMP shall manage its internal database for BL_KEY0, and shall be responsible for sending chipset vendors BL_KEY0, which will be embedded to designated chipset.

For BL_KEY1:

BL_KEY1 is owned by the operator. The key could be managed by either the operator or a thirt-party trusted authority, who signs the terminal software.

SDMP shall provide the method and procedure for signing BL_KEY1 with BL_KEY0. The owner of BL_KEY1 shall provide detailed information to SDMP, including chipset model and public key of BL_KEY1. SDMP shall sign BL_KEY1's public key with BL_KEY0's private key, and return the result together with BL_KEY1's public key to BL_KEY1's owner, so that BL_KEY1's public key and signed result can be preset into the terminal device.

A.5 Security requirements of the bootloader

The security requirements of the bootloader focus on the bootup and download process. Bootloader in flash should be copied to random access memory (RAM) before boot, and boot from RAM (BFR). The bootloader shall comply with the chain of trust mechanism to verify software signature during boot and loader.

For the bootup process:

The bootloader shall not launch subsequent component software until their signatures are verified. The bootloader shall verify the signature again every time the subsequent software restarts. Only the bootloader stored in the terminal's flash can be executed. The signature verifications and executions of all software shall be performed in RAM.

For the download:

The bootloader shall write downloaded software and its signature to flash only after the signature of the downloaded software is verified in RAM. After the software is successfully stored, the terminal shall perform a complete reboot. If the downloaded software exceeds the maximum size of the flash allocated by bootloader, the bootloader shall reject the software and reboot. Downgrades of upgradable software should be avoided.

A.6 Performance requirements of bootloader and terminal security chipset

To ensure terminal user experience, a terminal security chipset shall be used to provide hardware acceleration for hash algorithm.

Annex B

Two-way DCAS APIs

(This annex forms an integral part of this Recommendation.)

B.1 Java APIs

The standard Java virtual machine solution has been widely used in the industry for downloading and executing applications. Figure B.1 illustrates the runtime environment of the DCAS client software.

DCAS client software is an Xlet application running on terminal software platform that supports Java runtime environment.

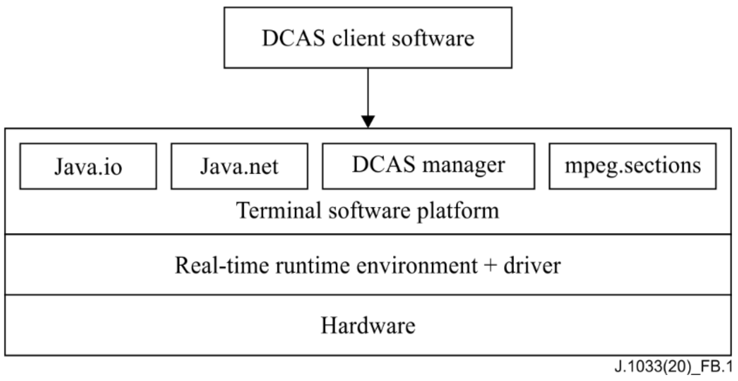

Figure B.1 – Diagram of DCAS JAVA APIs

B.1.1 APIs type

B.1.1.1 APIs for DCAS Manager

B.1.1.1.1 A DCAS module manager (DCAS Manager) has been defined for the terminal software platform to manage requests for descrambling services. DCAS Manager includes an upper layer API and bottom layer API of terminal software platform, and extension application APIs. The upper layer APIs of terminal software platform.

A CA module manager is defined by the upper-level APIs of terminal software platform, to manage requests for descrambling services (which means to descramble video/audio streams). A DCAS client software must register the CA module in CA module manager in order to receive descramble request from the terminal software platform on a terminal device.

The DCAS client software requires terminal software platform to implement the upper-layer APIs of the DCAS terminal software platform.

B.1.1.1.2 The bottom-layer APIs of the terminal software platform

Except for the existing Java APIs, DCAS client software requires a collection of Java APIs that are implemented on the terminal software platform, including the extension APIs required for accessing the terminal security chipset

B.1.1.1.3 Extension application APIs

By extending DCAS APIs, the conditional access system (CAS) management module on a terminal software platform can perform basic CA information communication with Java Apps without being limited by using IXC between Java application and DCAS application.

DCAS client software requires DCAS extension APIs to be implemented by terminal software platform for DCAS applications.

B.1.1.1.4 Detachable security device APIs

DCAS applications can communicate with detachable security devices via this APIs.

B.1.1.2 Network APIs

DCAS client software can use Java network APIs to access network resources, such as interconnection with headend CA server.

DCAS client software requires terminal software platform to implement the existing Java network API according to the definition in Java.net.

B.1.1.3 MPEG section filter APIs

DCAS client software uses MPEG Section Filter API to load MPEG Section for CA. CA related data includes ECM, EMM and conditional access table (CAT).

DCAS client software requires terminal software platform to implement MPEG Section Filter APIs according to definitions in org.davic.mpeg.sections, org.davic.mpeg.TransportStream and org.davic.net.tuning.NetworkInterface.

B.1.1.4 Non-volatile storage APIs

DCAS client software can use the existing Java APIs to access terminal storage, including storing data in non-volatile storage.

DCAS client software requires terminal software platform to implement non-volatile storage APIs.

DCAS client software can store data by using designated directory in the file system of a non-volatile storage. Terminal software platform needs to provide proper functions to read the name of root path of the file system.

B.1.2 APIs invoking sequence

This clause describes two scenarios where DCAS API is used: CASModule registration and channel switching. See Figure B.2 for CASModule registration and Figure B.3 for channel switching.

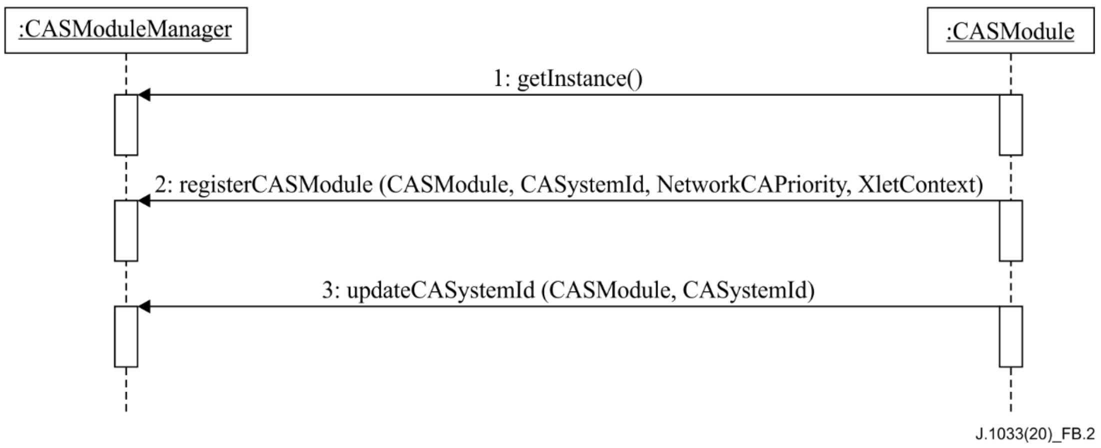
:CASModuleManager
:CASModule
1: getInstance()
2: registerCASModule (CASModule, CASystemId, NetworkCAPriority, XletContext)
3: updateCASystemId (CASModule, CASystemId)
J.1033(20)_FB.2

Figure B.2 – CA Module Registration in CASModuleManager

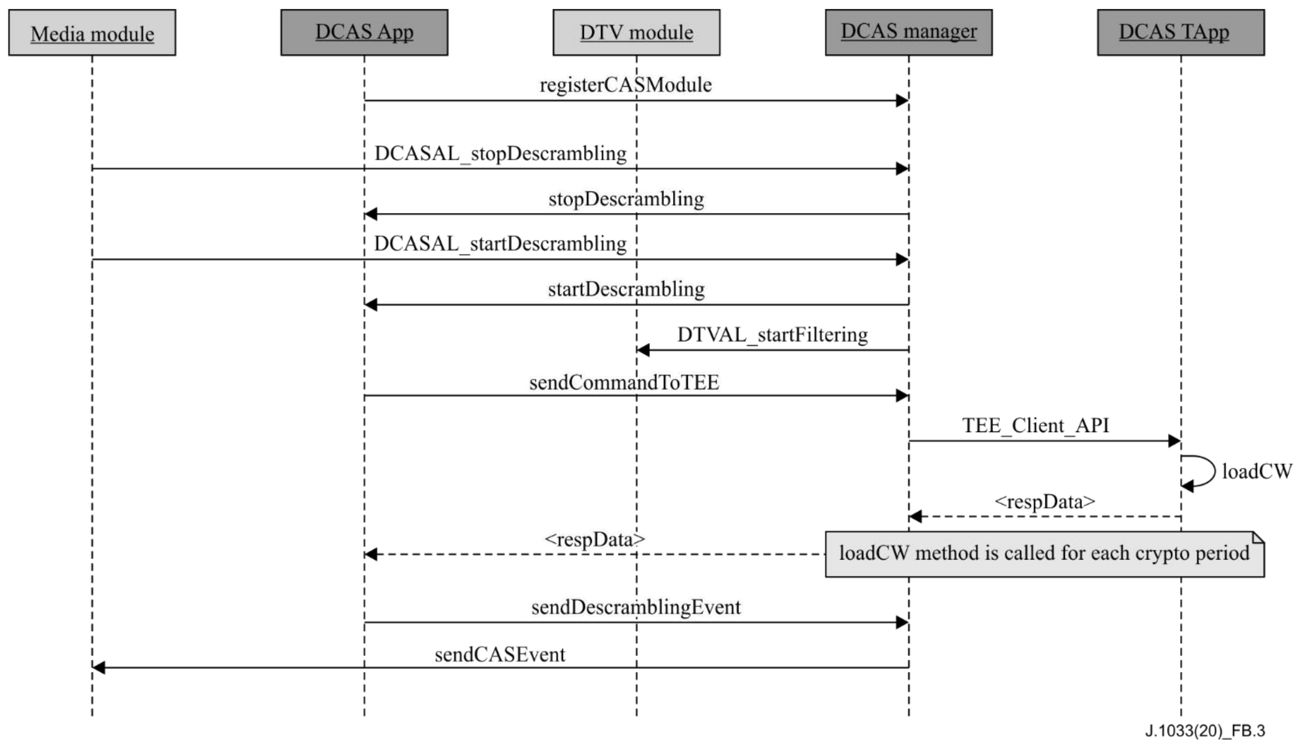

Figure B.3 – Channel Selection

B.1.3 APIs description

See Table B.1 for the names of the APIs.

Table B.1 – APIs

API name	Package name
Terminal software platform upper layer API	org.ngb.net.cas.module
Terminal software platform bottom layer API	org.ngb.net.cas.controller
Extension Application API	org.ngb.net.cas.event
Detachable Security Device API	org.ngb.net.cas.detachable
Network API	java.net
Section Filter API	org.davic.mpeg.sections
Non-volatile Storage API	java.io

B.1.4 Package org.ngb.net.cas.module

Packet org.ngb.net.cas.module provides the upper layer APIs of DCAS terminal software platform, which needs to be implemented in TVOS.

Refer to Table B.2 for the overview of Package org.ngb.net.cas.module

Table B.2 – Overview of package org.ngb.net.cas.module

Interface	
CASModule	To denote the CASModule object that requests descrambling of a group of elementary streams.
CASDataUtils	To obtain and set CA information, as well as read and write DCAS data.
CADescriptor	Provide information of CA descriptor. The program map table (PMT) of a given service may provide CA descriptors. CAT may also have CA descriptors.
CAServiceComponentInfo	For extracting component information of a CA service, for example, ECM packet identification (PID) and DescreamblerContext for loading CW.
CASPacketListener	DCAS application uses this interface to receive out of band CAS Packets (e.g., EMMs).
CASSession	Provide information about a CAS session.
CAStatus	DCAS application sends CAStatus every time the descrambling status in DescramblerContext changes. This status is used to indicate if descrambling is successful. If any component fails to be descrambled, this status must report the failure of the request on descrambling the entire service. When receiving a new CAStatus, terminal software platform shall use the CAS event described in this clause to notify other applications.
CATListener	Required by DCAS application to filter in-band EMM using CA descriptor in CAT.
CATNotifier	Used by DCAS application to register the listener for receiving CAT update notification.
Class	
CASModuleManager	Used to register all CASModules implemented by all DCAS applications.
CASPermission	Any DCAS application must get CASPermission to access CASModuleManager. This mechanism ensures only network operator authorised DCAS can use DCAS APIs.

B.1.4.1 Interface org.ngb.net.cas.module.CASModule

B.1.4.1.1 Methods

B.1.4.1.1.1 startDescrambling

Prototype:

public void startDescrambling (CASSession,
CAServiceComponentInfo[]casci)

Description:

This method is invoked by terminal software platform to request CASModule to descramble a group of elementary streams in a given session.
DCAS application gets the relevant Network Interface object from CAS session, and gets the TransportStream object from the NetworkInterface object, which will be used for ECM section filtering via org.davic.mpeg.sectionsAPI.

Parameters:

casSession: Session for the descrambling request.

casci: CA service component information array. This array can be used to get ECM PID, as well as for the DescramblerContext object for loading CW in DCAS.

Returns:

None.

B.1.4.1.1.2 updateDescrambling

Prototype:

```
public void updateDescrambling (  CASSession casSession,
                                  CAServiceComponentInfo[]casci )
```

Description:

invoked by terminal software platform to update the descrambling component list in CASModule. According to request, CASModule will start descrambling components added to the array and stop descrambling removed components.
There will be no change to components after update.
Note, this method is rarely invoked. It usually happens due to change of PMT in a session.
Terminal software platform can also invoke this method to notify CAModule when there is any change to a CAS session, e.g., the session type.

Parameters:

casSession: Session for the descrambling request.
casci: CA service component information array. This array can be used to get ECM PID, as well as for the DescramblerContext object for loading CW in DCAS.

Returns:

None.

B.1.4.1.1.3 stopDescrambling

Prototype:

```
public void stopDescrambling ( CASSession casSession )
```

Description:

Invoked by terminal software platform to request CASModule to stop descrambling all components in a given session.

Parameters:

casSession Session for the descrambling request.

Returns:

None.

B.1.4.1.1.4 getCAInfo

Prototype:

```
public String getCAInfo (  int cmdId,
                           String data)
```

Description:

Invoked by terminal software platform to get CA information.

Parameters:

cmdId: ID for command, can be extended according to actual project requirements.
data: Inquiry parameter.

Returns:

CA information data.

B.1.4.1.1.5 setCAInfo

Prototype:

```
public int setCAInfo ( int cmdId,
                       String data)
```

Description:

Invoked by terminal software platform to set CA information.

Parameter:

cmdId: Unique ID for command, can be extended according to actual project's requirements.
Data: CA information data

Returns:

Return 0 if the set is successful

B.1.4.2 Interface org.ngb.net.cas.module.CASDataUtils

B.1.4.2.1 Description

This interface is used to represent general API for getting CA information.

B.1.4.2.2 Methods

B.1.4.2.2.1 getCAInfo

Prototype:

```
public String getCAInfo (  int casId,
                           int cmdId,
                           String data )
```

Description:

Terminal software platform gets CA information from DCAS application in response to user operations.

Parameters:

casId: Identifier of a CAS vendor
cmdId: Unique ID for command, can be extended according to actual project requirements
data: inquiry data

Returns:

CA information data.

B.1.4.2.2.2 setCAInfo

Prototype:

```
public int setCAInfo (   int casId,
                         int cmdId,
                         String data )
```

Description:

Terminal software platform sets CA information for DCAS application in response to user operations.

Parameters:

casId: Identifier of a CAS vendor
cmdId: Unique ID for command, can be extended according to actual project requirements
data: CA information data

Returns:

Return 0 if set successfully

B.1.4.2.2.3 getData

Prototype:

```
public String getData (  int casId,
                                  int cmdId,
                                  int[] type )
```

Description:

Terminal software platform gets data from DCAS manager in response to user operations.

Parameters:

casId: Identifier of a CAS vendor

cmdId: Unique ID for command, can be extended according to actual project requirements

type: Data type reference

Returns:

Data obtained

B.1.4.2.2.4 setData

Prototype:

```
public int setData (  int casId,
                      int cmdId,
                      int type,
                      String data )
```

Description:

Terminal software platform sets CA information for DCAS application in response to user operations.

Parameters:

casId: Identifier of a CAS vendor.

cmdId: Unique ID for command, can be extended according to actual project requirements.

Data: DCAS data

Type: Data type

Returns:

Return 0 if set successfully

B.1.4.3 Interface org.ngb.net.cas.module.CADescriptor

B.1.4.3.1 Description

This interface provides information of CA descriptor, which may present in the PMT of a given service. CA descriptor may also appear in CAT.

B.1.4.3.2 Methods

B.1.4.3.2.1 getCASystemId

Prototype:

```
public int getCASystemId ( )
```

Description:

This method returns the CASystemId of inside CA descriptor.

Parameters:

None.

Returns:

CASystemId.

B.1.4.3.2.2 getPid

Prototype:

```
public int getPid ( )
```

Description:

This method returns PID (ECM PID or EMM PID) in CA descriptor.

Parameters:

None.

Returns:

PID value.

B.1.4.3.2.3 getPrivateData

Prototype:

public byte[] getPrivateData ( )

Description:

This method returns private data array in CA descriptor.

Parameters:

None.

Returns:

privateData array.

B.1.4.4 Interface org.ngb.net.cas.module.CAServiceComponentInfo

B.1.4.4.1 Description

This interface is used to extract information of certain CA service components, such as ECM PID and DescramblerContext for loading CW.

B.1.4.4.2 Methods

B.1.4.4.2.1 getDescramblerContext

Prototype:

public DescramblerContext getDescramblerContext ( )

Description:

This method returns the DescramblerContext object used by DCAS application for loading CW. In the cycle of PMT components, when the same CA descriptor (with same ECMPID and private data) appears multiple times there should be only one DescramblerContext object.

Parameters:

None.

Returns:

DescramblerContext object.

B.1.4.4.2.2 getCADescriptor

Prototype:

public CADescriptor getCADescriptor ( )

Description:

This method returns the CA descriptor related to service components. CADescriptor instance is generated by CA information in PMT.

Parameters:

None.

Returns:

A CADescriptor object.

B.1.4.4.2.3 getComponentStreamPIDs

Prototype:

public int[] getComponentStreamPIDs ( )

Description:

This method returns an elementary stream array described in PMT, the order of array elements shall be consistent with that of array elements returned by getComponentStreamType.

Parameters:

None.

Returns:

ES PID array.

B.1.4.4.2.4 getComponentStreamTypes

Prototype:

public int[] getComponentStreamTypes ( )

Description:

This method returns stream type array in PMT, stream types shall comply with MPEG standard. The order of array elements shall be in consistence with that of the array elements returned by getComponentStreamPID.

Parameters:

None.

Returns:

Stream Type Array.

B.1.4.4.2.5 getServiceIdentifiers

Prototype:

public int[] getServiceIdentifiers ( )

Description:

This method returns the service identifier array associated with an object, the form of service identifier is determined by the actual usage scenario.

Parameters:

None.

Returns:

ServiceID array.

B.1.4.5 Interface org.ngb.net.cas.module.CASPacketListener

B.1.4.5.1 Description

DCAS application uses this interface to receive out-of-band CAS Packets (e.g., EMMs). According to given CA system ID, DCAS application uses the method registerCasPacketListener provided by class CASModuleManager to register this listener. CA system ID is denoted as casId. Receiving of CAS packet depends on the implementations of terminal software platform.

B.1.4.5.2 Methods

B.1.4.5.2.1 casPacketArrived

Prototype:

```
public void casPacketArrived (   int casId,
                                 byte [] casPacketData,
                                 byte [] casPacketHeader )
```

Description:

DCAS application uses the registered listener to get CAS packets.

Parameters:

casId CA: CA system ID

casPacketData: CAS packet data

casPacketHeader: Terminal software platform dependent CAS packet header.

Returns:

None.

B.1.4.6 Interface org.ngb.net.cas.module.CASSession

B.1.4.6.1 Description

This interface provides information for CAS session.

B.1.4.6.2 Constants – Session Types

B.1.4.6.2.1 TYPE_PRESENTATION

public static final int TYPE_PRESENTATION = 0x00000001

B.1.4.6.2.2 TYPE_RECORDING

public static final int TYPE_RECORDING =   0x00000002

B.1.4.6.2.3 TYPE_BUFFERING

public static final int TYPE_BUFFERING =   0x00000004

B.1.4.6.3 Methods

B.1.4.6.3.1 getType

Prototype:

public int getType ( )

Description:

This method returns the session type of a session.

Parameters

None.

Returns:

Session type, which can be one or a combination of values defined in this interface.

For example – a result of 0x00000003 returned of this method is a combination of type (0x00000001) and (0x00000002)

B.1.4.6.3.2 getNetworkInterface

Prototype:

public org.davic.net.tuning.NetworkInterface getNetworkInterface ( )

Description:

This method returns the NetworkInterface associated with CAS session. DCAS application can get NetworkInterface object from a CAS session. Using NetworkInterface, DCAS application can get object TransportStream, for invoking the org.davic.mpeg.sections APIs to perform ECM Section filtering.

Paramerters:

None.

Returns:

The NetworkInterface object.

B.1.4.6.3.3 getAssociatedService

Prototype:

public java.lang.Object getAssociatedService ( )

Description:

This method returns the service associated with a CAS session.

Parameters:

None.

Returns:

A Service object.

B.1.4.6.3.4 getServiceContext

Prototype:

public java.lang.Object getServiceContext ( )

Description:

This method returns ServiceContext associated with a CAS session.

NOTE – This method returns null in some cases where ServiceContext has no actual meaning.

Parameters:

None.

Returns:

ServiceContext object.

B.1.4.7 Interface org.ngb.net.cas.module.CAStatus

B.1.4.7.1 Description

Prototype:

public interface CAStatus

Description:

DCAS application uses this interface when invoking the sendDescramblingEvent method in CASModuleManager. DCAS application sends CAStatus every time the descrambling status in DescramblerContext changes. This status is used to indicate success or failure of a descrambling. If any one of the descrambling components fails to be descrambled, this status must report the failure of the descrambling request on the entire service. When terminal software platform receives a new CAStatus, it should notify other applications the CAS event described in this clause. Detailed usages of this interface are described in the Extension Application Interface clause.

B.1.4.7.2 Methods

B.1.4.7.2.1 isSuccess

Prototype:

public boolean isSuccess ( )

Description:

This method returns status of a descrambling request.

Parameters:

None.

Returns:

Return true if descrambling succeeds, or false if descrambling fails.

B.1.4.7.2.2 getCAToken

Prototype:

public int getCAToken ( )

Description:

This method returns parameters with which other applications to inquire network information from DCAS application via IXC.

Parameters:

None.

Returns:

CA token.

B.1.4.8 Interface org.ngb.net.cas.module.CATListener

DCAS application requires this interface to filter in-band EMM using the CA descriptor in CAT. DCAS application requires registerCATListener defined in CATNotifier to register the listener.

B.1.4.8.1 Methods

B.1.4.8.1.1 catUpdate

Prototype:

```
public void catUpdate ( CADescriptor desc,
                        org.davic.net.tuning.NetworkInterface ni)
```

Description:

This interface is used for notifying DCAS application of CAT update at specified network interface. DCAS application can get the TransportStream object with the NetworkInterface object.

EMM section filtering can be achieved by TransportStream object by calling org.davic.mpeg.clauses APIs. Client Software Platform will notify any CAT update to the CAT listener registered with the corresponding casId.

NOTE – If CAT is no longer to be filtered (after being successfully filtered); or CA descriptor is deleted from CAT, terminal software platform should invoke as catUpdate (null, theNetworkInterface).

Parameters:

Desc: The CA descriptor. DCAS application use the CASDescriptor object to get EMM PID

ni: Updated NetworkInterface where CAT is transported

Returns:

None.

B.1.4.9 Interface org.ngb.net.cas.module.CATNotifier

B.1.4.9.1 Description

Prototype:

```
public interface CATNotifier
```

Description:

DCAS application uses this interface to register the listener for CAT update notification.

DCAS application uses CAT information to filter in-band EMM.

B.1.4.9.2 Methods

B.1.4.9.2.1 registerCATListener

Prototype:

```
public void registerCATListener (  int casId,
                                   CATListener catListener )
```

Description:

DCAS application invokes this method to register a CATListener.

Parameters:

casId: CA system ID

catListener: CATListerner object for registration.

Returns:

None.

B.1.4.9.2.2 unregisterCATListener

Prototype:

```
public void unregisterCATListener ( CATListener catListener )
```

Description:

DCAS application invokes this method to unregister a CATListener.

Parameters:

catListener: CATListener that needs to be unregistered.

Returns:

None.

B.1.4.10 Class org.ngb.net.cas.module.CASModuleManager

B.1.4.10.1 Description

Used to register all CASModules implemented by DCAS applications.

B.1.4.10.2 Methods

B.1.4.10.2.1 getInstance

Prototype:

public static CASModuleManager getInstance ( ) throws java.lang.SecurityException

Description:

This method is used to get a CASModuleManager instance.

Parameters:

None.

Returns:

CASModuleManager Instance.

Exception Handling:

java.lang.SecurityException – Throw this exception when security policy is enforced but no org.ngb.net.ca.module.CASPermission has been assigned to the invoker.

B.1.4.10.2.2 registerCASmodule

Prototype:

```
public void registerCASModule (CASModule caModule,
                               int caSystemId,
                               int networkCAPriority,
                               java.lang.Object context )
                              throws java.lang.IllegalArgumentException
```

Description:

This method is used by DCAS application for registering a CASModule on terminal software platform.

Parameters:

caModule: CASModule that needs to be registered.

caSystemId: caSystemId managed by the CASmodule.

networkCAPriority: used when registering more than one CASModules in CASModuleManager. Operator can decide whether this parameter is optional for each CASModule. When the priority policy is enforced, operator needs to specify priority for every CASModule. Terminal software platform should send descrambling request to the registered CASModule which has the hightest priority and of which the managed caSystemId mapping to the CA descriptor can be found in PMT. When the priority policy is disabled, DCAS application should set this parameter to 0. The method for selecting CASModule will be determined by the Terminal software platform.

context: context of DCAS application that needs to register a CASModule. Used for terminal software platform in deciding to identify of a DCAS application.

Returns:

None.

Exception Handling:

java.lang.IllegalArgumentException – Throw this exception if the specified CASModule instance has been registered.

B.1.4.10.2.3 updateCASystemId

Prototype:

public void updateCASystemId (CASModule aModule,

int caSystemId)
throws java.lang.IllegalArgumentException

Description:

This method is used by DCAS application to update CASystemId in a CASModule on application platform.

Parameters:

aModule : CASModule to be updated

caSystemId : new caSystemId to be associated to the CASModule.

Returns:

None.

Exception handling:

java.lang.IllegalArgumentException: If given CASModule instance has not been registered.

B.1.4.10.2.4 sendDescramblingEvent

Prototype:

```
public void sendDescramblingEvent ( CASModule aModule,
                                    CASSession casSession,
                                    CAStatus aCAStatus )
                                  throws java.lang.IllegalArgumentException
```

Description:

This method is used by DCAS App to return a CAStatus to terminal software platform. Every time a DescramblerContext changes as any scrambled element in a service changes, DCAS App must send CAStatus to indicate if the descrambling is successful. If any element fails to descramble, CAStatus must notify the descrambling failure of the entire service.

When receiving a new CAStatus, terminal software platform should continue to send the information to related applications using CAS Event defined in extension Application APIs.

Parameters:

aModule: Associated CASModule.

casSession: Session for descrambling request.

aCAStatus: CAStatus to be sent.

Returns:

None.

Exception Handling:

java.lang.IllegalArgumentException: if the given CASModule instance has not been registered.

B.1.4.10.2.5 unregisterCASModule

Prototype:

```
public void unregisterCASModule (CASModule aModule) throws java.lang.IllegalArgumentException
```

Description:

This method is used by DCAS application to unregister a CASModule from terminal software platform.

Parameters:

aModule: CASModule to be unregistered.

Returns:

None.

Exception Handling:

java.lang.IllegalArgumentException: if the given CASModule instance has not been registered.

B.1.4.10.2.6 getChipControllers

Prototype:

```
public ChipController[] getChipControllers ( )
```

Description:

This method is used by DCAS application to request available chip controller list from terminal software platform. This method returns one chip controller for each terminal security chipset. Most terminals support only one chip controller, in which case the returned array only contains one element.

Parameters:

None.

Returns:

A chip controller array.

B.1.4.10.2.7 setCurrentController

Prototype:

```
public void setCurrentController (  CASModule aModule,
                                     ChipController aChipController )
                                  throws ava.lang.IllegalArgumentException
```

Description:

This method is used to set up default chip controller according to given CAModule for descrambling. Not required to specify it in CASModuleManager if this method is not invoked.

Parameters:

aModule: Associated CASModule.

aChipController: Default chip controller in use.

Returns:

None.

Exception Handling:

java.lang.IllegalArgumentException: If the given CASModule instance has not been registered.

B.1.4.10.2.8 setCCIBits

Prototype:

```
public void setCCIBits ( CASModule aModule,
                         CASSession casSession,
                         int cciBits )
```

Description:

This method is used for setting up CCI (Copy Control Information) bits required for copy control of a service. The definition of CCI bits is assigned, and will be explained and executed by terminal software platform.

Parameters:

aModule: Associated CASModule.

casSession: Session for descrambling request.

cciBits: CCI bits value used by current service.

Returns:

None.

B.1.4.10.2.9 setServiceListFilter

Prototype:

```
public void setServiceListFilter ( int filterData )
```

Description:

This method is used to provide terminal software platform parameters for filtering service list.

Definition of service list parameter is assigned and executed by terminal software platform.

Parameters:

filterData: filterData service list filter parameters.

Returns:

None.

B.1.4.10.2.10 registerCASPacketListener

Prototype:

```
public void registerCASPacketListener (  int casId,
                                         CASPacketListener casPacketListener )
                                        throws java.lang.IllegalArgumentException
```

Description:

This method is used by DCAS application to register a CASPacketListener. The CASPacketListener is invoked by terminal software platform to transport CAS data packet (e.g., EMM) to DCAS application. CA system ID is denoted by the casID parameter. The retrieving of CAS data packet is implemented by terminal software platform itself.

Parameters:

casId: CasystemID

casPacketListener: CASPacketListener to be registered

Returns:

None.

Exception Handling:

java.lang.IllegalArgumentException: If a listener has been registered with the given casID.

B.1.4.10.2.11 unregisterCASPacketListener

Prototype:

```
public void unregisterCASPacketListener ( CASPacketListener casPacketListener )
                                         throws java.lang.IllegalArgumentException
```

Description:

This method is used by DCAS application to unregister a CASPacketListener.

Parameters:

casPacketListener: The listener to be unregistered.

Returns:

None.

Exception Handling:

java.lang.IllegalArgumentException: If the given CASPacketListener has not been registered.

B.1.4.10.2.12 getDetachableSecurityDevices

Prototype:

```
public DetachableSecurityDevice[] getDetachableSecurityDevices ( )
```

Description:

This method is used by DCAS application to get the object handle of a detachable device (e.g., smart card).

Parameters:

None.

Returns:

A DetachableSecurityDevice object.

B.1.4.10.2.13 receiveOsdMsg

Prototype:

```
public void receiveOsdMsg(byte[] msg, int[] flags)
```

Description:

Display on-screen display (OSD) message, the meaning of its parameters are project-specific.

Parameters:

msg: OSD message content, it can also contain descriptive information other than text content.

flags: OSD type indicator.

Returns:

None.

B.1.4.10.2.14 showFingerMsg

Prototype:

```
public void showFingerMsg ( CASModule aModule,
                            CASSession casSession,
                            byte[] msg )
```

Description:

Display fingerprint message, the meaning of its parameters are project-specific.

Parameters:

aModule: Associated CASModuel.

casSession: Session for descrambling request.

msg: Fingerprint information, NULL for disabling fingerprint display.

Returns:

None.

B.1.4.10.2.15 receiveTuningAlert

Prototype:

```
public void receiveTuningAlert ( int[] serviceIdentifiers,
                                 int[] flags)
```

Description:

Emergency broadcast. In some projects the parameters of an emergency broadcast are not sent by CA system, in which case this function is not required to be implemented.

Parameters:

serviceIdentifiers: A group of values for identifying emergency broadcast channel parameters. Meaning of the values are defined in actual project.

Flags: Parameters used to denote the type of emergency broadcast.

Returns:

None.

B.1.4.10.2.16 getCATNotifier

Prototype:

```
public CATNotifier getCATNotifier ( )
```

Description:

This method is used by DCAS application to get the CATNotifier object. DCAS application can register to the CAT notifier with a listener for getting CAT update notification

Parameters:

None.

Returns:

The CAT Notifier object.

B.1.4.11 Class org.ngb.net.cas.module.CASPermission

B.1.4.11.1 Description

Prototype:

```
public class CASPermission extends java.security.BasicPermission
```

Description:

Any DCAS application must get CASPermission before accessing CASModuleManager.

This mechanism is used to ensure only the DCAS App authorized by network operator is able to use DCAS APIs.

B.1.4.11.2 Methods

B.1.4.11.2.1 CASPermission

Prototype:

```
public CASPermission ( String name )
```

Description:

Create a new CASPermission. If the Name string is not to be used and it should be set to NULL.

Parameters:

Name: Name of the CASPermission.

Returns:

None.

B.1.4.11.2.2 CASPermission

Prototype:

```
public CASPermission ( String name,
                                     String actions )
```

Description:

Create a new CASPermission. If the Name string is not to be used and it should be set to NULL. If the actions string is not to be used and it should be set to NULL. This constructor is used by the java.security.Policy object to instantiate a new Permission object.

Parameters:

Name: Name of the CASPermission.

Actions: Action list.

Returns:

None.

B.1.5 Package org.ngb.net.cas.controller

The org.ngb.net.cas.controller package provides the bottom APIs for DCAS terminal software platform, which needs to be implemented in TVOS.

See Table B.3 for the overview of org.ngb.net.cas.controller package.

Table B.3 – Overview of org.ngb.net.cas.controller package

Interface	
DescramblerContext	Component for controlling the descrambling function of terminal security chipset. Multiple DescramblerContext instances can be used to descramble multiple streams using different keys.
ChipController	Component used to control the execution of terminal security chipset.
Class	
Key	A fundmental cryptographic key, used to describe the cryptography used by K-LAD and the output parameters of the cipher function.
CWKey	Descrambling key, a.k.a. control word.

B.1.5.1 Interface org.ngb.net.cas.controller.DescramblerContext

B.1.5.1.1 Description

Prototype:

public interface DescramblerContext

Description:

Component used to control the descrambling of terminal security chipset. Multiple DescramblerContext instances can be used to descramble multiple streams using different keys.

B.1.5.1.2 Methods

B.1.5.1.2.1 loadCW

Prototype:

```
public void loadCW (int Vendor_SysID,
                    CWKey cwKey,
                    Key[] levelKeys,
                    int schemeId )
                  throws CADriverException
```

Description:

This method is used to load CW to descrambler on terminal software platform, and load required keys into terminal security chipset.

A descrambler channel is a logical collective of all streams descrambled by a single CW.

It depends on DescreamblerContext to use the scrambler channel.

Besides, DCAS App should notify terminal software platform that current CW is invalid (e.g., due to unauthorized entitlements), and terminal software platform should stop the corresponding descrambling. In this case, DCAS App will provide a null CWKey.

Parameters:

Vendor_SysID: This value is used to identify CA vendor and support root key derivation in controller. The root key of terminal security chipset is derived from this value.

cwKey: control word. If control word is plaintext, the levelKeys parameter is ignored. If cwKey is null, no valid control word is provided by DCAS App.

levelKeys: Multi-level keys to be set to terminal security chipset. The index of key array is the same as its absolute position in terminal security chipset. In an array an element with value Null indicates that no key should be loaded to the corresponding place in terminal security chipset.

So: levelKey[0] is Key 1 (encrypted by Key 2); levelKey[1] is Key 2 (encrypted by Key3); levelKey[2] is not to be used.

schemeId: This schemeId is used to specify cryptography for terminal security chipset. A list of scheme values are defined in the ChipController interface. This value is ignored if controller only supports one scheme.

Returns:

None.

Exception Handling:

CADriverException: If key loading fails.

B.1.5.1.2.2 overrideChipController

Prototype:

```
public void overrideChipController ( ChipController aChipController )
                                    throws CADriverException
```

Description:

This method is used by DCAS App to request terminal software platform to override default terminal security chipset key ladder (can be done by invoking the setCurrentController method of CASModuleManager). If this method is not invoked, terminal security chipset will use the default controller. This method is only used in terminal security chipset systems where multiple terminal security chipsets are implemented.

Parameters:

aChipController: Controller to be overridden.

Returns:

None.

Exception Handling:

CADriverException: If operation fails.

B.1.5.2 Interface org.ngb.net.cas.controller.Chipcontroller

B.1.5.2.1 Description

Prototype:

```
public interface ChipController
```

Description:

component used to control the execution of terminal security chipset.

B.1.5.2.2 Constants

B.1.5.2.2.1 SCHEME_TDES

public static final int SCHEME_TDES=0

Description: Used to indicate triple-DES (TDES) should be used by terminal security chipset. TDES is defined in [NIST 3DES].

B.1.5.2.2.2 SCHEME_AES

public static final int SCHEME_AES=1

Description: Used to indicate AES should be used by terminal security chipset.

B.1.5.2.2.3 SCHEME_SM4

public static final int SCHEME_SM4=2

Description: Used to indicate SM4 may be used by terminal security chipset.

B.1.5.2.2.4 PROCESSING_MODE_REGULAR

public static final int PROCESSING_MODE_REGULAR=0

Description: Used to indicate that terminal security chipset doesn't need additional processing in the challenge-response algorithm.

B.1.5.2.2.5 PROCESSING_MODE_POST_PROCESSING

Public static final int PROCESSING_MODE_POST_PROCESSING=1

Description: Used to indicate that terminal security chipset needs post processing in the challenge-response algorithm.

B.1.5.2.3 Methods

B.1.5.2.3.1 getPublicId

Prototype:

public byte[] getPublicId ( ) throws CADriverException

Description:

This method returns the public ID of terminal security chipset.

Parameters:

None.

Returns:

The publicId of terminal security chipset.

Exception Handling:

CADriverException: If communication error occurs when accessing the driver of terminal security chipset.

B.1.5.2.3.2 getChipType

Prototype:

public byte[] getChipType ( ) throws CADriverException

Description:

This method returns the type ID of terminal security chipset.

Parameters:

None.

Returns:

Terminal security chipset type.

Exception Handling:

CADriverException: if communication error occurs when accessing the driver of terminal security chipset

B.1.5.2.3.3 getChipControllerProperty

Prototype:

```
public java.lang.String getChipControllerProperty ( java.lang.String propertyName )
                                                    throws CADriverException
```

Description:

This method returns the value of the corresponding property according to the property name provided for terminal security chipset. This function is reserved in this interface, and can be used to read properties that will be added to the controller in the future. No property name is defined at present.

Parameters:

propertyName: Property name.

Returns:

Property value.

Exception Handling:

CADriverException: if communication error occurs when accessing the driver of terminal security chipset

B.1.5.2.3.4 authenticate

Prototype:

```
public byte[] authenticate (  int Vendor_SysID,
                               byte[] challenge,
                               Key[] levelKeys,
                               int schemeId,
                               int processingMode )
                             throws CADriverException
```

Description:

This method is used to authenticate the key ladder mechanism in terminal security chipset. Terminal security chipset should calculate authentication information according to the input challenge information.

Parameters:

Vendor_SysID: This value is used to identify CA vendor, which is used to support root key derivation in controller. The root key of terminal security chipset is derived from this value.

challenge: Challenge information, nonce.

levelKeys: level keys required by key ladder. The index of key array is equal to the absolute position in terminal security chipset. In an array an element with value Null indicates that no key should be loaded to the corresponding place in terminal security chipset. That is, levelKey[0] is Null; levelKey[1] is Key 2 (encrypted by Key 3); levelKey[2] is not used.

schemeId: This schemeId is used to specify the cipher algorithm of terminal security chipset (such as TDES, AES and SM4). The ChipController interface defines the list of scheme. If the controller supports only one scheme, then the value will be ignored.

processingMode: it is used to specify whether additional post-processing in the calculation of the response needs to be implemented. If the controller only supports no post-processing mode, then the parameter will be ignored.

Returns:

Response calculated by terminal security chipset.

Exception Handling:

CADriverException: If communication error occurs when accessing the driver of terminal security chipset.

B.1.5.2.3.5 encryptData

Prototype:

```
public void encryptData (  int Vendor_SysID,
                           CWKey cwKey,
                           Key[] levelKeys,
                           int schemeId,
                           int encryptionId,
                           byte[] src,
                           int srcPos,
                           byte[] dest,
                           int destPos,
                           int length )
                         throws CADriverException
```

Description:

This method invokes chip functions to encrypt data in memory.

Parameters:

Vendor_SysID: this parameter is used to identify CA vendor. Security chipset uses this value to derive root key.

cwKey: control word for encryption. If control word is not encrypted, subsequent levelKeys will be ignored.

levelKeys: The index of key element in the array is equal to the absolute position in key ladder. Null element in array indicates no key need to be set in the corresponding position in the key ladder.

schemeId: Cipher algorithm used by key ladder. If the chipset supports only one algorithm, then the parameter will be ignored.

encryptionId: Algorithm of data encryption/decryption, if the chipset supports only one algorithm, then the parameter will be ignored.

src: source data array.

srcPos: starting position of source data array.

dest: Destination data array.

destPos: starting position of destination data array.

length: length of data to be processed, in byte.

Exception Handling:

CADriverException: Throw the CADriverException exception when key ladder communication error occurs.

B.1.5.2.3.6 decryptData

Prototype:

```
public void decryptData (  int Vendor_SysID,
                              CWKey cwKey,
                              Key[] levelKeys,
                              int schemeId,
                              int encryptionId,
                              byte[] src,
                              int srcPos,
                              byte[] dest,
                              int destPos,
                              int length)
                            throws CADriverException
```

Description:

This method invokes chipset functions to decrypt data in memory.

Parameters:

Vendor_SysID: This parameter is used to identify CA vendor. Security chipset uses this value to derive root key.

cwKey: Control word for decryption. If control word is not encrypted, subsequent levelKeys will be ignored.

levelKeys: The index of key element in the array is equal to the absolute position in key ladder. Null element in array indicates no key need to be set in the corresponding position in the key ladder.

schemeId: Cipher algorithm used by key ladder. If the chipset supports only one algorithm, then the parameter will be ignored.

encryptionId: Algorithm of data encryption/decryption. If the chipset supports only one algorithm, then the parameter will be ignored.

Src: source data array.

srcPos: starting position of source data array.

dest: destination data array.

destPos: starting position of destination data array.

length: data size to be processed, in byte.

Returns:

None.

Exception Handling:

CADriverException: Throw the CADriverException exception when key ladder communication error occurs.

B.1.5.3 Class org.ngb.net.cas.controller.Key

B.1.5.3.1 Description

Prototype:

public class Key

Description:

It denotes a basic cipher key used to describe the cryptography for K-LAD and output parameters of cipher functions.

B.1.5.3.2 Methods

B.1.5.3.2.1 Key

Prototype:

```
public Key ( byte[] value,
             boolean encrypted )
```

Parameters:

Value: The key value.

Encrypted: Tag to indicate if a key is encrypted. True means key has been encrypted, false means key is plaintext.

B.1.5.3.2.2 getKeyValue

Prototype:

public byte[] getKeyValue ( )

Description:

This method returns the key value.

Parameters:

None.

Returns:

The key value.

B.1.5.3.2.3 isEncrypted

Prototype:

public boolean isEncrypted ( )

Description:

When this method returns true, it means key is encrypted, whereas false means key is not encrypted.

Parameter:

None.

Returns:

True means key is encrypted, false means key is not encrypted.

B.1.5.4 Class org.ngb.net.cas.controller.CWKey

B.1.5.4.1 Description

Prototype:

public class CWKey extends Key

Description:

This class denotes descrambling key or control word.

B.1.5.4.2 Constant

B.1.5.4.2.1 PARITY_EVEN

public static final int PARITY_EVEN = 0

B.1.5.4.2.2 PARITY_ODD

public static final int PARITY_ODD = 1

B.1.5.4.3 Methods

B.1.5.4.3.1 CWKey

Prototype:

```
public CWKey (byte[] value,
              boolean encrypted,
              int parity)
```

Description:

value: value of key.

encrypted: true indicates key is encrypted, false indicates key is not encrypted.

parity: indicates the parity of control word.

B.1.5.4.3.2 getParity

Prototype:

public int getParity ( )

Description:

This method returns the parity of control word.

Parameters:

None.

Returns:

The parity of control word.

B.1.5.5 Class org.ngb.net.cas.controller.CASTEEManager

B.1.5.5.1 Description

Prototype:

public class CASTEEManager

Description:

The interface for communicating with TA in TEE.

B.1.5.5.2 Methods

B.1.5.5.2.1 sendCommandToTEE

Prototype:

```
public byte[] sendCommandToTEE (  byte[] teeAppUUID,
                                  int commandId,
                                  byte[] inputData )
                                throws CADriverException
```

Description:

DCAS App selects a dedicated security application, and send data to it.

Parameters:

teeAppUUID: UUID of the selected TAPP.

commandId: Type of command.

inputData: Data input.

Returns:

Data returned.

Exception Handling:

CADriverException: throws CADriverException if communication error occurs in the interaction with TEE driver.

B.1.6 Package org.ngb.net.cas.event

The org.ngb.net.cas.event package provides extension APIs package for DCAS. It's required to implement this package by DCAS of TVOS.

See Table B.4 for overview of Org.ngb.net.cas.event package.

Table B.4 – Overview of org.ngb.net.cas.event package

Interface	
CASEventListener	Shall be implemented by the application that needs to receive CAS event.
CASAppInfo	Provides information of DCAS App
CASEventInfo	Provides information of CASEvent
Class	
CASEventManager	CASEventManager should be used to register listener to get CAS event.

B.1.6.1 Interface org.ngb.net.cas.event.CASEventListener

B.1.6.1.1 Description

Prototype:

public interface CASEventListener

Description:

This interface should be implemented by the application that needs to receive CAS events. CAS events provide CA Status and basic information of the current ServiceContext.

B.1.6.1.2 Methods

B.1.6.1.2.1 receiveCASEvent

Prototype:

```
public void receiveCASEvent (  Object serviceContext,
                               int appId, int orgId,
                               boolean isSuccess,
                               int caToken )
```

Description:

This method is used to transfer CAS event to App that has registered CAS event listener.

Parameter:

serviceContext: Handle to which CAS event belongs.

appId: Used to identify the DCAS App that sends events. The Id can be used by App to communicate with DCAS App via IXC. When no DCAS App is available to descramble a given stream, terminal software platform should use NULL as the value of appId to invoke this method. Application for receiving notifications about such CAS event should handle such case according to its design and implementation.

orgId: orgId is used to identify the organization of DCAS App that sends events.

isSuccess: Bolean value used to indicate descrambling success/failure.

caToken: Token sent back to DCAS via IXC. Applications can use this token to search specific network information via IXC.

B.1.6.1.2.2 receiveCASOSDEvent

Prototype:

```
public void receiveCASOSDEvent ( Object serviceContext,
                                 int appId,
                                 int orgId,
                                 byte[] msg,
                                 int[] flag )
```

Description:

This method is used to transfer OSD event of CAS to application that has registered CAS event listener.

Parameters:

serviceContextCAS: Handle to which CAS OSD event belongs.

appId: Used to identify the DCAS App that send events. These ID can be used by App to communicate with DCAS App via IXC. When no DCAS App is available to descramble given stream, terminal software platform should use NULL as the value of casAppId to invoke this method. Application for receiving notfications about such CAS event should handle such case according to its design and implementation.

orgId: orgId is used to identify the organization of DCAS App that sends events.

msg: Used to transfer OSD content.

flag: Used to identify OSD type.

B.1.6.1.2.3 receiveCASFingerEvent

Prototype:

```
public void receiveCASFingerEvent ( Object serviceContext,
                                    int appId,
                                    int orgId,
                                    byte[] msg)
```

Description:

This method is used to transfer CAS fingerprint event to application that has registered CAS event listener.

Parameters:

serviceContext: Handle to which CAS Fingerprint event belongs.

appId: Used to identify the DCAS App that send events. These ID can be used by App to communicate with DCAS App via IXC. When no DCAS App is available to descramble given stream, terminal software platform should use NULL as the value of casAppId to invoke this method. Application for receiving notifications about such CAS event should handle such case according to its design and implementation.

orgId: orgId is used to identify the organization of DCAS App that send events.

msg: Used to transfer the fingerprint data

B.1.6.2 Interface org.ngb.net.cas.event.CASAppInfo

B.1.6.2.1 Description

Prototype:

```
public interface CASAppInfo
```

Description:

This interface provides information of DCAS application.

B.1.6.2.2 Methods

B.1.6.2.2.1 getAID

Prototype:

public int getAID()

Description:

This method returns the application ID of DCAS application.

Parameters:

None.

Returns:

The application ID of DCAS application.

B.1.6.2.2.2 getOID

Prototype:

public int getOID ( )

Description:

This method returns the organization ID of DCAS application.

Parameters:

None.

Returns:

The organization ID of DCAS application.

B.1.6.3 Interface org.ngb.net.cas.event.CASEventInfo

B.1.6.3.1 Description

Prototype:

public interface CASEventInfo

Description:

This interface provides information of CASEvent.

B.1.6.3.2 Constant

B.1.6.3.2.1 TYPE_PRESENTATION

public static final int TYPE_PRESENTATION = 0x00000001

B.1.6.3.2.2 TYPE_RECORDING

public static final int TYPE_RECORDING =   0x00000002

B.1.6.3.2.3 TYPE_BUFFERING

public static final int TYPE_BUFFERING =   0x00000004

B.1.6.3.3 Methods

B.1.6.3.3.1 getType

Prototype:

public int getType()

Description:

This method returns the type of the operation that produces the CAS Event.

Parameters:

None.

Returns:

Operation type, can be one or combination of the values defined in this interface.

For example – A returned value 0x00000003 is combination of type (0x00000001) and (0x00000002)

B.1.6.3.3.2 getNetworkInterface

Prototype:

public org.davic.net.tuning.NetworkInterface getNetworkInterface ( )

Description:

This method returns the NetworkInterface related to CAS Event.

Parameters:

None.

Returns:

A NetworkInterface object.

B.1.6.3.3.3 getAssociatedService

Prototype:

public java.lang.Object getAssociatedService ( )

Description:

This method returns the associated service to the CAS Event.

Parameters:

None.

Returns:

A Service object.

B.1.6.3.3.4 getServiceContext

Prototype:

public java.lang.Object getServiceContext ( )

Description:

This method returns associated ServiceContext to the CAS Event.

Please note that in some cases, ServiceContext may not have actual meaning, and this method thus returns null.

Parameters:

None.

Returns:

A ServiceContext object.

B.1.6.4 Class org.ngb.net.cas.event.CASEventManager

B.1.6.4.1 Description

Prototype:

public class CASEventManager

Description:

Application uses CASEventManager to register listener to receive CAS event.

CA event provides current CA Status and basic information.

B.1.6.4.2 Methods

B.1.6.4.2.1 getInstance

Prototype:

public static CASEventManager getInstance ( )

Description:

This method is used to get a CASEventManager instance. Singleton.

Parameters:

None.

Returns:

The CASEventManager instance.

B.1.6.4.2.2 addListener

Prototype:

public void addListener (CASEventListener aCASEventListener)

Description:

This method is used by application to register a CASEventListener for transferring all CAS events.

Parameters:

aCASEventListener: CASEventListener to be registered

Returns:

None.

B.1.6.4.2.3 removeListener

Prototype:

public void removeListener (CASEventListener aCASEventListener)

Description:

This method is used to unregister a CASEventListener.

Parameters:

aCASEventListener: CASEventListener a registered CASEventListener.

Returns:

None.

B.1.7 Package org.ngb.net.cas.detachable

The org.ngb.net.cas.detachable package provides DCAS detachable security device APIs. TVOS needs to implement this package.

See Table B.5 for the overview of Org.ngb.net.cas.detachable package.

Table B.5 – Overview of Org.ngb.net.cas.detachable package

Interface	
DetachableSecurityDevice	Used by application to register the listener for detachable security device to get the plugging status of a device.
DetachableSecurityDeviceListener	The listener for detachable security device status. It should be implemented by the application that needs to listen to the plugging status of a device.

B.1.7.1 Interface DetachableSecurityDevice

B.1.7.1.1 Description

This interface denotes the components used to control communications with detachable security devices (e.g., smart card).

B.1.7.1.2 Methods

B.1.7.1.2.1 open

Prototype:

public void open ( ) throws CADriverException

Description:

This method is used by DCAS App to initiate session with detachable security devices.

Parameters:

None.

Returns:

None.

Exception Handling:

CADriverException: If driver error occurs.

B.1.7.1.2.2 close

Prototype:

public void close ( ) throws CADriverException

Description:

This method is used by DCAS application to close session with detachable security devices.

Parameters:

None

Returns:

None.

Exception Handling:

CADriverException: If driver error occurs.

B.1.7.1.2.3 reset

Prototype:

public byte[] reset ( ) throws CADriverException

Description:

This method is used to reset detachable security device and return data (ATR in the case of smart card).

Parameters:

None.

Returns:

A byte array to store the data returned after device reset.

Exception Handling:

CADriverException: If driver error occurs.

B.1.7.1.2.4 sendData

Prototype:

public void sendData (byte [] data) throws CADriverException

Description:

This method is used by DCAS App to send data to detachable security device.

Parameter:

data: Data to be sent (the command APDU in the case of smart card)

Returns:

None.

Exception Handling:

CADriverException: if driver error occurs.

B.1.7.1.2.5 registerListener

Prototype:

public void registerListener (DetachableSecurityDeviceListener aListener)

Description:

This method is used by DCAS to register the listener for receiving data sent by detachable security device.

Parameters:

aListener: DetachableSecurityDeviceListener to be registered

Returns:

None.

B.1.7.1.2.6 removeListener

Prototype:

public void removeListener ( )

Description:

This method is used by DCAS App to remove a registered listener.

Parameters:

None.

Returns:

None.

B.1.7.2 Interface DetachableSecurityDeviceListener

B.1.7.2.1 Description

This method should be implemented by DCAS App to receive the status of detachable security device and data sent by it.

B.1.7.2.2 Constant

B.1.7.2.2.1 DEVICE_IN

public static final int DEVICE_IN = 1

Description: used to describe status of detachable security device: Inserted (indicates smart card is inserted in the case of smart card)

B.1.7.2.2.2 DEVICE_OUT

public static final int DEVICE_OUT = 2

Description: used to describe status of detachable security device: unplugged (indicates smart card is unplugged in the case of smart card)

B.1.7.2.2.3 DEVICE_ERROR

public static final int DEVICE_ERROR = 3

Description: used to describe status of detachable security device: ERROR (indicates smart card error in the case of smart card)

B.1.7.2.3 Methods

B.1.7.2.3.1 receiveDeviceStatus

Prototype:

public void receiveDeviceStatus (int status)

Description:

This method should be implemented by DCAS App to receive status of detachable security device.
Notify DCAS App when status of detachable security device changes.

Parameters:

status: Status of detachable security device (See description of the field).

Returns:

None.

B.1.7.2.3.2 receiveData

Prototype:

public void receiveData (byte [] data)

Description:

This method is invoked when detachable security device sends data to DCAS App.

Parameters:

data: Data sent by detachable security device (response APDU in the case of smart card)

Returns:

None.

B.2 Javascript APIs

B.2.1 Overview

DCAS Client Software can be developed based on DCAS Javascript APIs, in order to run in a client software platform that supports HTML5 execution environment. See Table B.6 for the overview of DCAS Javascipt Interface.

Table B.6 – DCAS App Javascript Interface Overview

Class Name	Description
JSDCAS.CASDescriptor	The CA descriptor object represents a CAS Descriptor which appears either in the PMT (Program Map Table) for a selected MPEG service, or in the CAT (Conditional Access Table).
JSDCAS.CASEcmEvent	The ECM event object contains ECM event.
JSDCAS.CASEmmEvent	The EMM Event object contains EMM event.
JSDCAS.CASFilter	The CAS Filter object represents filter criteria that the JSDCAS application defines when requesting the platform to filter OOB EMMs or inband EMMs.
JSDCAS.CASM	This is the CASM global object by which DCAS manager object and controller object can be accessed.
JSDCAS.CASModule	An interface of a CAS Module object that the JSDCAS application should implement and export by registering in the platform CAS Module Manager, in order to receive descrambling requests, ECMs, EMMs, and any other metadata that the specific JSDCAS application requires.
JSDCAS.CASModuleManager	CAS Module Manager utilized by the JSDCAS application to receive descrambling requests, ECMs, EMMs, as well as report CAS descrambling status.
JSDCAS.CASPacketEvent	The CAS Packet Event object notify JS DCAS application about any CAS packet event.
JSDCAS.CASSession	The object represents a CAS Session that is generated by the platform for a specific descrambling request.
JSDCAS.CASStatus	This object is created by the JSDCAS application for reporting a CAS Status to the platform DCAS manager.
JSDCAS.TeeController	The TEE Controller object is used by the JSDCAS application to communicate with the TEE (Trusted Execution Environment).
JSDCAS.TeeRetVal	The TEE Ret object is returned from TEE containing data or error.

B.2.2 API calling sequence

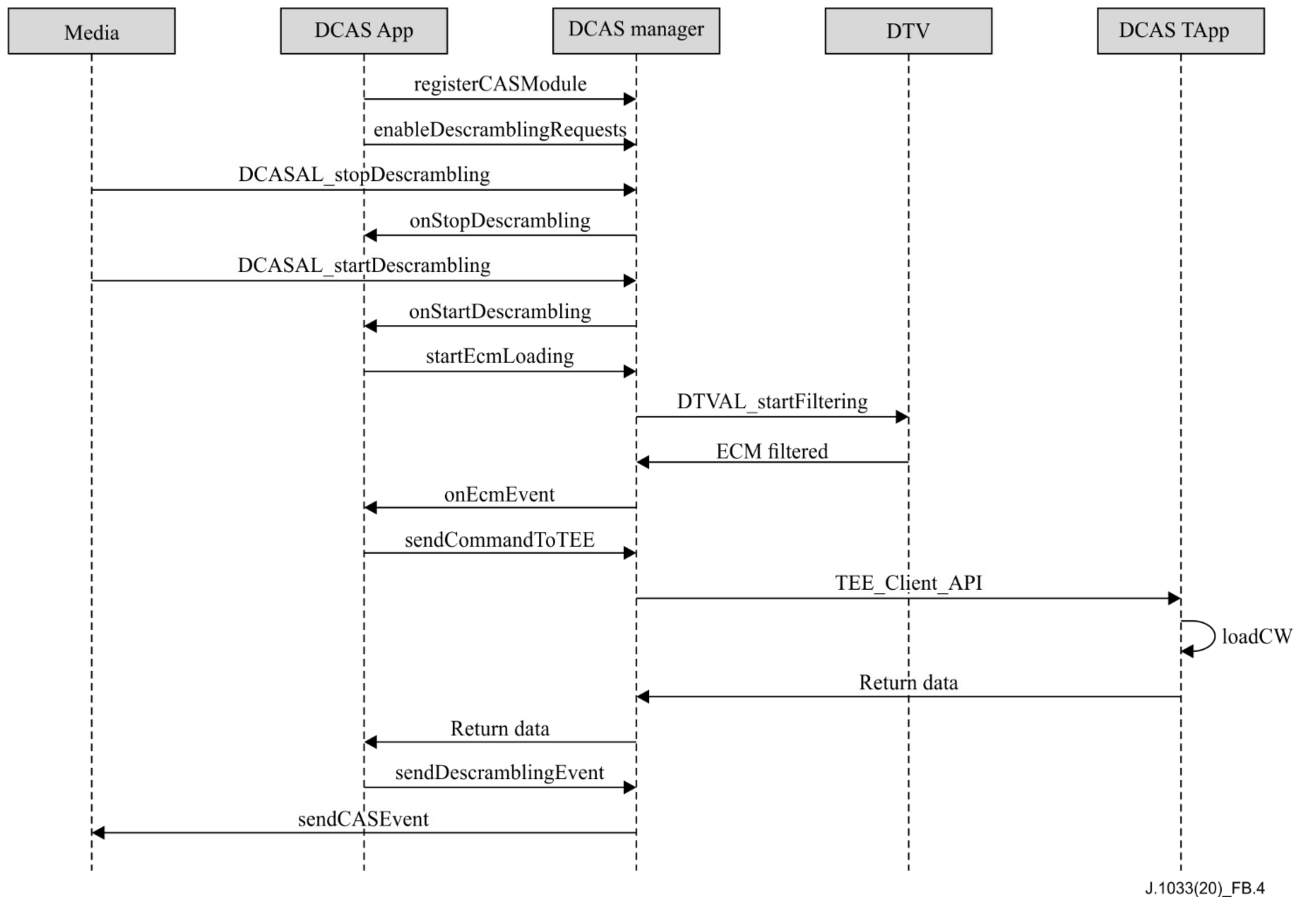

Figure B.4 – Basic Calling Sequence of DCAS APIs

B.2.3 Class JSDCAS.CASDescriptor

This object is used to represent CA descriptors in PMT or CAT.

B.2.3.1 getCasId

Prototype:

{number}getCasId ( )

Description:

This method is provided by terminal software platform and returns the CAS ID that appears in the CAS Descriptor.

B.2.3.2 getPid

Prototype:

{number}getPid ( )

Description:

This method returns the PID that appears in the CAS Descriptor. The PID can be either ECM PID (if the CAS descriptor appeared in the PMT), or EMM PID (if the CAS descriptor appeared in the CAT).

B.2.3.3 getPrivateData

Prototype:

{Uint8Array}getPrivateData ( )

Description:

This method returns the private data that appears in the CAS Descriptor. The private data is returned in the format of Uint8Array.

B.2.4 Class JSDCAS.CASEcmEvent

This class contains the information that can be received within an ECM event. An ECM event is passed to the JSDCAS application via the method CASModule.onEcmEvent or via the method CASModuleManager.onStartDescrambling when auto-load first ECM feature is enabled. The event can represent ECM packet arrivals, or timeout, or notifications of internal errors in the section filtering process in the device.

B.2.4.1 getEcmData

Prototype:

{Uint8Array}getEcmData ( )

Description:

Returns complete ECM data, or Null for a timeout event or internal error.

B.2.4.2 getError

Prototype:

{number}getError ( )

Description:

This method returns the error value reported by the section filtering process. Used only for debug. This method can only be invoked when the event does not provide any other information.

B.2.4.3 getTableId

Prototype:

{number} getTableId ( )

Description:

This method returns the Table ID of the ECM packet that received. This method can only be invoked when the event can provide ECM data.

B.2.4.4 isTimeout

Prototype:

{boolean} isTimeout ( )

Description:

Returns the value to indicate if there is a timeout for receiving the first ECM packet. The timeout duration can be set via CASModuleManager.enableDescramblingRequests API. This method can only be invoked when event does not provide ECM data.

Returns:

Ture – There is timeout.

False – No timeout.

B.2.5 Class JSDCAS.CASEmmEvent

This class contains the information of EMM event.

An EMM event is passed to the JSDCAS application via the method CASModule.onInbandEmmEvent.

The event may represent a CAT arrival on the tuned transport stream, or for notification of inband EMM packet arrival, or for notification of internal error in the section filtering process in the device.

B.2.5.1 getEmmData

Prototype:

{Uint8Array}getEmmData ( )

Description:

This method returns full EMM data. If the event is notified by CAT update or internal error, this method returns null.

B.2.5.2 getError

Prototype:

{number}getError ( )

Description:

This method returns the error value reported by the section filtering process. Used only for debug.

This method should be called only when the event does not provide any other information.

B.2.5.3 getTableId

Prototype:

{number} getTableId ( )

Description:

Return the Table ID of the EMM packet.

B.2.5.4 isCatUpdateNotification

Prototype:

{boolean}isCatUpdateNotification ( )

Description:

This method indicates whether the EMM event received is due to CAT update or not.

In case of CAT update, the EMM data shall be null.

Returns:

True – when there is CAT update.

False – the event is notified by EMM arrival, or an internal error.

B.2.6 Class JSDCAS.CASFilter

The CAS Filter object represents filter criteria that the JSDCAS application defines when requesting the platform to filter OOB EMMs or inband EMMs. The platform should invoke the CAS Module only when packets match the filtering rules. Any packet that does not match the filter rules should be discarded by the platform, and the application framework should not be invoked. A CASFilter (or an array of CAS filters) can be set in the methods CASModuleManager.startCasPacketLoading and CASModuleManager.startInbandEmmLoading.

The filter criteria includes:

a) An offset (in byte) from the beginning of the packet. All the bytes before the offset are always ignored and are not part of the comparison rules.

b) A bitmap, used to compare with the coming packet.

c) A bitmap mask representing which bits locations should be included during the comparison in b)

   For every bit that is set to '0' in this mask, this specific bit location should be ignored during the comparison in b) above.

B.2.6.1 getBitmapMask

Prototype:

{Uint8Array}getBitmapMask ( )

Description:

Returns the bitmap mask.

B.2.6.2 getBitmapValue

Prototype:

{Uint8Array}getBitmapValue ( )

Description:

Returns the bitmap value for comparison.

B.2.6.3 getOffset

Prototype:

{number}getOffset ( )

Description:

Returns offset (in bytes).

B.2.7 Class JSDCAS.CASM

CASM is a global object that can be used to access all DCAS manager and controller objects.

B.2.7.1 getCASModuleManager

Prototype:

{JSDCAS.CASModuleManager}getCASModuleManager ( )

Description:

Returns an object instance of the CAS Module Manager.

B.2.7.2 getTeeController

Prototype:

{JSDCAS.TeeController}getTeeController ( )

Description:

Returns an object instance of the TEE Controller.

B.2.8 Class JSDCAS.CASModule

An interface of a CAS Module object that the JSDCAS application should implement and register in the platform CAS Module Manager, in order to receive descrambling requests, ECMs, EMMs, and any other metadata a specific JSDCAS application requires.

B.2.8.1 getCasId

Prototype:

{number}getCasId ( )

Description:

This method returns the unique CAS ID associated with this CAS module. The JSDCAS application must implement this method to return a valid CAS ID, before calling CASModuleManager.registerCASModule. This value is expected to appear in the CA descriptors within PMT for scrambled services as well as appear in the CA descriptor within CAT.

B.2.8.2 onCasPacketEvent

Prototype:

onCasPacketEvent (casPacketEvent)

Description:

This method is called by the CAS Module Manager platform to notify the JSDCAS application when an out-of-band EMM or any other out-of-band CAS packet is received by the device.

See the method CASModuleManager.startCasPacketLoading for more descriptions.

Parameters:

CASPacketEvent casPacketEvent: This parameter is a CASPacketEvent instance that contains the received OOB EMMs or other out-of-band CAS packets.

B.2.8.3 onEcmEvent

Prototype:

```
onEcmEvent (  casSession,
              ecmEvent )
```

Description:

This method is called by the platform DCAS Manager to notify the JSDCAS application when a new ECM is filtered. In fast mode, the platform will invoke this method after setting the CW carried in ECM in K-LAD.

Parameters:

CASSession casSession – The CAS Session object returned by CASModule.onStartDescrambling.
CASEcmEvent ecmEvent – a CASEcmEvent instance that contains ECM.

B.2.8.4 onInbandEmmEvent

Prototype:

```
onInbandEmmEvent (  casSessionForEMM,
                    emmEvent )
```

Description:

This method is called by the platform DCAS Manager to notify the JSDCAS application when a new inband EMM is filtered or when a CAT is filtered and updated on the tuned transport stream.

Parameters:

CASSession casSessionForEMM: A special CAS Session object representing the tuner and the transport stream in which the CAT is found, and including the CAS Descriptor of the CAT (instead of CAS Descriptor of the PMT).
The platform should create a dedicated CAS Session (with different session ID) for this purpose. Note that in this case, the CAS Session object may be filled only partially and may not contain all service information within.
NOTE – If the CAS descriptor with the matching CAS ID is removed from the CAT on that specific transport stream, or the STB/device is tuned out of the transport stream, there shall still be a CAT update notification but the CAS Session shall be null.
CASEmmEvent emmEvent: CASEmmEvent object that contains the EMM, or the CAT notification, or any error.

B.2.8.5 onStartDescrambling

Prototype:

```
onStartDescrambling (  casSession,
                       firstEcmEvent)
```

Description:

This method is called by the platform DCAS Manager to invoke the JSDCAS application with a new descrambling request. It is usually called when platform tunes and starts to descramble a new channel.
The JSDCAS application can receive this descrambling requests only after it called CASModuleManager.enableDescramblingRequests.
In the case of auto-load where the First ECM feature is enabled (in CASModuleManager.enableDescramblingRequests), the platform automatically starts filtering for the first ECM, and invokes this method with a valid CASECMEvent as the second parameter. In this case, the JSDCAS application should not call CASModuleManager.startEcmLoading explicitly.
This method can be called several times simultaneously if there are several tuners in the device and each one may be used to tune to different service (or even if it is the same service). Each tuning triggers its own descrambling request with a corresponding CAS session.
Another case that this method can be called several times simultaneously if the stream components of the service are not scrambled in the exact same way, whether they use different ECMs, or different

private data in their respective CAS descriptors in the PMT. Each request will have its own CAS Session.

Parameters:

CASSession casSession: CAS Session objects generated by the platform for specific descrambling request. Each object has unique session ID and all information about service and elementary streams, as well as the CA descriptor for this CAS ID in PMT.

CASEcmEvent firstEcmEvent: The first ECM packet (or timeout or error) the platform receives in auto-load mode. Set it as Null when it is not in auto-load mode.

B.2.8.6 onStopDescrambling

Prototype:

onStopDescrambling (casSession)

Description:

This method is called by the platform DCAS Manager to notify the JSDCAS application to stop an ongoing descrambling session. This usually happens when the device tunes out of the scrambled channel, before it tunes to a new channel.

Parameters:

CASSession casSession-The CAS Session object returned by CASModule.onStartDescrambling.

B.2.9 Class JSDCAS.CASModuleManager

This is the platform CAS Module Manager utilized by the JSDCAS application to receive descrambling requests, ECMs, EMMs, as well as to report CAS descrambling statuses. The JSDCAS application shall implement a CAS Module object and register it as a listener in the platform CAS Module Manager.

B.2.9.1 Enums

JSDCAS.CASModuleManager.ACTION_ERROR_ACTION_NOT_SUPPORTED

JSDCAS.CASModuleManager.ACTION_ERROR_DRIVER

JSDCAS.CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS

JSDCAS.CASModuleManager.ACTION_ERROR_NETWORK

JSDCAS.CASModuleManager.ACTION_ERROR_SECURITY

JSDCAS.CASModuleManager.ACTION_OK

JSDCAS.CASModuleManager.PROP_ID_BOUQUET

JSDCAS.CASModuleManager.PROP_ID_CAS_VENDOR_ID

JSDCAS.CASModuleManager.PROP_ID_CAS_VERSION

JSDCAS.CASModuleManager.PROP_ID_CHIP_ID

JSDCAS.CASModuleManager.PROP_ID_USER_BITS

JSDCAS.CASModuleManager.PROP_ID_SECURE_BITS

JSDCAS.CASModuleManager.PROP_ID_STB_ACTIVE_STATUS

JSDCAS.CASModuleManager.PROP_ID_ZIPCODE

JSDCAS.CASModuleManager.PROP_TYPE_NUMBER

JSDCAS.CASModuleManager.PROP_TYPE_STRING

JSDCAS.CASModuleManager.PROP_TYPE_UINT8ARRAY

B.2.9.2 Methods

B.2.9.2.1 disableDescramblingRequests

Prototype:

{number}disableDescramblingRequests (casModule)

Description:

This method is called by the JSDCAS application to stop receiving descrambling requests via the CAS Module. It is called in rare cases when JSDCAS wants temporarily not to receive requests, or wants to re-configure the work mode parameters, or before manual shutdown.

A call to CASManager.enableDescramblingRequests will renew the reception of descrambling requests.

Parameters:

CASModule casModule – Instance of CAS module.

Returns:

Success – CASModuleManager.ACTION_OK.

Failure -Returns the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS -Invalid parameters.

CASModuleManager.ACTION_ERROR_DRIVER -Driver error.

B.2.9.2.2 enableDescramblingRequests

Prototype:

```
{number} enableDescramblingRequests (   casModule,
                                         firstEcmTimeout,
                                         autoLoadFirstEcm,
                                         isFastMode,
                                         ecmTableIds )
```

Description:

This method is called by the JSDCAS application to start receiving descrambling requests. Via this method, several parameters can be configured that set the mode of work between the platform DCAS Manager and the specific CAS Module. This method is usually called only once (after the CAS Module has been registered), as it is assumed that a specific JSDCAS application does not change the mode of work later.

For calling this method again with different configuration, the JSDCAS application should call CASModuleManager.disableDescramblingRequests firstly, to discard the current configuration.

Parameters:

CASModule casModule – Instance of CAS module.

Number firstEcmTimeout – The maximum length of time (in millisecond) the platform waits for the first ECM. If it times out, CAS module will invoke onEcmEvent or onStartDescrambling to receive CASEcmEvent.

boolean autoLoadFirstEcm -To specify if auto-load mode is to be used. In auto-load mode, after invoking this method, the platform automatically filters the first ECM without having to wait for JS DCAS App to invoke startEcmLoading.

boolean isFastMode -Fast mode (as placeholder, no actual meaning)

Array ecmTableIds -If JS DCAS App needs to specify the tableID for ECM.

Returns:

Success – CASModuleManager.ACTION_OK.

Failure – Return the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – Invalid parameters.

CASModuleManager.ACTION_ERROR_ACTION_NOT_SUPPORTED – specific mode not supported.

CASModuleManager.ACTION_ERROR_DRIVER – Driver error.

B.2.9.2.3 fetchDataFromCasHeadend

Prototype:

```
{Uint8Array|number} fetchDataFromCasHeadend (casModule,
                                              inputData,
                                              casHeURI )
```

Description:

By invoking this method, JS DCAS Application fetches data from headend via the platform, GPRS, or other possible methods in the future.

Parameters:

CASModule casModule-Instance of CAS module.

Uint8Array inputData – Data to be sent to headend.

String casHeURI – URI of headend server.

Returns:

Success – Data returned from headend.

Failure – Returns the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – Invalid parameters.

CASModuleManager.ACTION_ERROR_DRIVER – Driver error.

CASModuleManager.ACTION_ERROR_ACTION_NOT_SUPPORTED – method not supported.

CASModuleManager.ACTION_ERROR_NETWORK – network error.

B.2.9.2.4 registerCASModule

Prototype:

```
{number}registerCASModule (vendorId,
                           casModule,
                           networkPriority,
                           applicationContext)
```

Description:

JS DCAS App registers itself to the platform CAS Module Manager by using this method.

Parameters:

number vendorId – Vendor Id of CAS. Each CAS vendor has a unique specific ID.

CASModule casModule – CAS module instance to be registered.

Number networkPriority – If more than one CASModule is registered within the CASModuleManager, this value indicates the priority of CASModule. The value is defined by operator. A higher value means higher priority (e.g., 3 is higher priority than 2).

If priority is enforced by the network operator, the platform shall send descrambling request to the CASModule with the highest priority.

If priority is not enforced by the network operator, each JSDCAS application must pass a zero value for this parameter. In this case, which CASModule receives the descrambling request is depends on the platform's implementation.

* applicationContext – An additional platform-specific application parameter. It is usually passed to the application from the platform during initialization time. The use of this parameter is project-specific.

Returns:

Success – CASModuleManager.ACTION_OK.

Failure – Returns the following error values:

CASModuleManager.ACTION_ERROR_SECURITY – The caller application is not permitted to access this function in the CAS Module Manager.

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – Invalid parameter.

B.2.9.2.5 removeCASModule

Prototype:

```
{number}removeCASModule (vendorId,
                         casModule,
                         applicationContext)
```

Description:

This method is used for removing a registered CAS Module from the platform CAS Module Manager. It is called in rare cases when the JSDCAS application wants to change casId, or before manual shutdown.

Parameter:

number vendorId – Vendor Id of CAS. Each CAS vendor has a unique specific ID.

CASModule casModule – Instance of CAS module to be unregistered

* applicationContext – An additional platform-specific application parameter. It is usually passed to the application from the platform during initialization time. The use of this parameter is project-specific.

Returns:

Success – CASModuleManager.ACTION_OK.

Failure – Returns the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS -Invalid parameter.
CASModuleManager.ACTION_ERROR_DRIVER -Driver error.
CASModuleManager.ACTION_ERROR_SECURITY -Permission denied.

B.2.9.2.6 sendCommandToSTB

Prototype:

```
{number}sendCommandToSTB (casModule,
                          inputData)
```

Description:

Data channel function invoked by JS DCAS App to send data to DCAS Manager. DCAS Manager forwards commands to corresponding modules to process. The commands including OSD, upgrade trigger, fingerprint, emergency broadcast and audience survey, etc., which are sent by a business operations support system (BOSS). DCAS as data channel is only responsible for redistributing the commands.

Parameters:

CASModule casModule -Instance of CAS module registered.

Uint8Array inputData -Data to be sent to DCAS manager.

Returns:

Success – CASModuleManager.ACTION_OK.

Failure – Returns the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – Invalid parameter.
CASModuleManager.ACTION_ERROR_DRIVER – Driver error.
CASModuleManager.ACTION_ERROR_ACTION_NOT_SUPPORTED – Method not supported.
CASModuleManager.ACTION_ERROR_NETWORK – Network error.

B.2.9.2.7 sendDataToHeadend

Prototype:

```
{number}sendDataToHeadend (casModule,
                           inputData)
```

Description:

By invoking this method, JS DCAS sends data to headend via platform, GPRS or other possible methods in the future.

Parameters:

CASModule casModule -Instance of CAS module registered.

Uint8Array inputData -Data to be sent to headend.

Returns:

success – CASModuleManager.ACTION_OK.

Failure – Returns the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – Invalid parameter.
CASModuleManager.ACTION_ERROR_DRIVER – Driver error.
CASModuleManager.ACTION_ERROR_ACTION_NOT_SUPPORTED – Method not supported.

CASModuleManager.ACTION_ERROR_NETWORK – Network error.

B.2.9.2.8 sendDescramblingEvent

Prototype:

```
{number}sendDescrambingEvent ( casModule,
                               casSession,
                               casStatus )
```

Description:

This method is called by the JSDCAS application to report a CAS Status to the platform DCAS Manager. The JSDCAS application shall send a CAS Status to the platform DCAS Manager each time the descrambling status is changed within the session.

Parameters:

CASModule casModule – Instance of CAS module registered

CASSession casSession – CAS Session obtained from CASModule.onStartDescrambling.

CASStatus casStatus – CASStatus object generated by JS DCAS App.

Returns:

Success – CASModuleManager.ACTION_OK.

Failure – Returns the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – Invalid parameters.

CASModuleManager.ACTION_ERROR_ACTION_NOT_SUPPORTED – Method not supported.

B.2.9.2.9 sendFreeTextOSD

Prototype:

```
{number}sendFreeTextOSD ( casModule,
                          inputData,
                          flags )
```

Description:

This method is called by JSDCAS application to pass broadcasted free text to Middleware. The Middleware may pass the text to a UI application or choose to handle the OSD by itself, depends on the project requirements.

Parameters:

CASModule casModule – Instance of CAS module registered.

Uint8Array inputData – Text information

ArrayBuffer flags – Additional information indicating display method or format, etc. Project specific.

Returns:

Success – CASModuleManager.ACTION_OK.

Failure – Returns the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – Invalid parameters.

CASModuleManager.ACTION_ERROR_ACTION_NOT_SUPPORTED – Method not supported.

B.2.9.2.10 setCCIBits

Prototype:

```
{number}setCCIBits ( casModule,
                     casSession,
                     cciBits)
```

Description:

Set data bits of CCI (Copy Control Information)

Parameters:

CASModule casModule -Instance of CAS module registered.

CASSession casSession – CAS Session obtained from CASModule.onStartDescrambling.

Number cciBits – CCI bits.

Returns:

Success – CASModuleManager.ACTION_OK.

Failure – Returns the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – Invalid parameters.
CASModuleManager.ACTION_ERROR_DRIVER – Driver error.
CASModuleManager.ACTION_ERROR_ACTION_NOT_SUPPORTED – Method not supported.

B.2.9.2.11 setData

Prototype:

```
{number}setData ( casModule,
                  propertyId,
                  propertyType,
                  propertyValue )
```

Description:

For DCAS App to set platform properties including BouquetID, activation status, CAS information, Beidou information, ChipID, CASVendorID, region code and CA version, etc.

Parameters:

CASModule casModule – Instance of CAS module registered.
Number propertyId – Property ID, see JSDCAS.CASModuleManager.PROP_ID_xxx.
Number propertyType – Property Type, see JSDCAS.CASModuleManager.PROP_TYPE_xxx.
Number|string|Uint8Array propertyValue -Property Value.

Returns:

Success – CASModuleManager.ACTION_OK.

Failure – Returns the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS -Invalid parameters.
CASModuleManager.ACTION_ERROR_ACTION_NOT_SUPPORTED-Method not supported.

B.2.9.2.12 setPinCode

Prototype:

```
{number}setPinCode (casModule,
                    pinCode )
```

Description:

Set personal identification number (pin) code to platform.

Parameter:

CASModule casModule – instance of CAS module
Number pinCode -PIN code to be set.

Returns:

Success – CASModuleManager.ACTION_OK.

Failure – Returns the following error value:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – invalid parameters.
CASModuleManager.ACTION_ERROR_DRIVER – Driver error.
CASModuleManager.ACTION_ERROR_ACTION_NOT_SUPPORTED – Method not supported.

B.2.9.2.13 setServiceListFilter

Prototype:

```
{number}setServiceListFilter ( casModule,
                               filterData )
```

Description:

Set filtering criteria for service list. Definition of the filter criteria is platform specific.

Parameters:

CASModule casModule – Instance of CAS module.

Number filterData -Filtering criteria.

Returns:

Success – CASModuleManager.ACTION_OK.

Failure – Returns the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – Invalid parameters.

CASModuleManager.ACTION_ERROR_ACTION_NOT_SUPPORTED – Method not supported.

B.2.9.2.14 startCasPacketLoading

Prototype:

```
{number}startCasPacketLoading ( casModule,
                                cableModemFilter,
                                sourceURL,
                                casFilter )
```

Description:

This method is called by JS DCAS application to start receiving CAS packets from out of band. CAS packets can be EMMs or any other out-of-band metadata required by the JSDCAS application. The mechanism for receiving of the packets depends on the device hardware, the platform, and the network environment.

In devices which includes a Cable Modem, the reception of the CAS packets can be done via ADSG or BDSG protocol, or by using the implementation of IP over Cable to join a multicast IP address. Other devices like IPTVs can connect to Ethernet/Wi-Fi and can also join a multicast IP address to receive CAS packets.

There is also an option to tell the platform to open a User Datagram Protocol (UDP) datagram socket locally (on localHost 127.0.0.1) and bind it to specific local port.

In all cases – when new CAS packet arrives, the JSDCAS application can handle it via CASModule.onCasPacketEvent.

NOTE – Some platform implementations may support receiving of packets from more than one source simultaneously. In these cases, this function may be called more than once with different URLs or with different CAS Tunnel IDs.

Parameters:

CASModule casModule – Instance of CAS module.

Number|string cableModemFilter – In case of Cable Modem and DSG tunnel, a filter must be provided: In case DSG is not used, this parameter should be null.

String sourceURL – When opening a local UDP datagram socket, the string value of the URL should be as the following example:

"udp://@127.0.0.1:4444" or "udp://@localhost:4444" where 4444 is the local port that the socket should be bind to.

CASFilter|Array casFilter – Filter criteria, can be a CASFilter array.

Returns:

Success - CASModuleManager.ACTION_OK.

Failure – Returns the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – Invalid parameters.

CASModuleManager.ACTION_ERROR_DRIVER – Driver error.

CASModuleManager.ACTION_ERROR_ACTION_NOT_SUPPORTED – Method not supported.

B.2.9.2.15 startEcmLoading

Prototype:

```
{number}startEcmLoading (  casModule,
                           casSession)
```

Description:

This method is called by JSDCAS application to start receiving ECMs for a specific scrambled service. It is called after receiving descrambling request via the CAS Module with a valid CASSession object that is generated by the platform DCAS Manager. NOTE – In the case that auto-load first ECM feature is enabled, calling this method is not required. For more information about setting the auto-load first ECM feature, see the method CASModuleManager.enableDescramblingRequests.

Parameters:

CASModule casModule – Instance of CAS module.

CASSession casSession – CAS Session obtained from CASModule.onStartDescrambling.

Returns:

Success – CASModuleManager.ACTION_OK.

Failure – Returns the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS -Invalid parameters.
CASModuleManager.ACTION_ERROR_DRIVER -Driver error.

B.2.9.2.16 startInbandEmmLoading

Prototype:

```
{number}startInbandEmmLoading ( casModule,
                                emmTableIds,
                                casFilter,
                                includeCatNotifications)
```

Description:

This method can be called by JSDCAS application either to start receiving inband EMM, or to receive CAT as necessary. JSDCAS application receives data via CASModule.onInbandEmmEvent when there is EMM or CAT update.

Parameters:

CASModule casModule – Instance of CAS module.

Array emmTableIds – EMM table ID array.

CASFilter|Array casFilter – The platform will notify application only for data that matches the criteria. It is also possible to pass a Filter array.

boolean includeCatNotifications – Specifies whether the JSDCAS application also wishes to receive CAT update notification.

Returns:

Success – CASModuleManager.ACTION_OK.

Failure – Return the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS -Invalid parameters.
CASModuleManager.ACTION_ERROR_DRIVER -Driver error.
CASModuleManager.ACTION_ERROR_ACTION_NOT_SUPPORTED -Method not supported.

B.2.9.2.17 stopCasPacketLoading

Prototype:

```
{number}stopCasPacketLoading ( casModule,
                               cableModemFilter,
                               sourceURL)
```

Description:

JS DCAS App invokes this method to stop receiving out-of-band CAS data packet.

Parameters:

CASModule casModule – Instance of CAS module.

Number|string cableModemFilter -Required by Cable Modem.

String sourceURL – Required when receiving by UDP.

Returns:

Success – CASModuleManager.ACTION_OK.

Failure – Return the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS -Invalid parameters.
CASModuleManager.ACTION_ERROR_DRIVER -Driver error.

B.2.9.2.18 stopEcmLoading

Prototype:

{number}stopEcmLoading (casModule,
casSession)

Description:

This method is called by JSDCAS application to temporarily stop receiving ECMs. JS DCAS App rarely invokes this method. CASManager.startEcmLoading needs to be re-invoked to resume ECM receiving.

Parameters:

CASModule casModule – Instance of CAS module.

CASSession casSession – CAS Session obtained from CASModule.onStartDescrambling.

Returns:

Success – CASModuleManager.ACTION_OK.

Failure – Returns the following values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS -Invalid parameters.
CASModuleManager.ACTION_ERROR_DRIVER -Driver error.

B.2.9.2.19 stopInbandEmmLoading

Prototype:

{number}stopInbandEmmLoading (casModule)

Description:

This method is called by JS DCAS App to stop receiving in band EMM. JS DCAS App rarely invokes this method. CASManager.startInbandEmmLoading needs to be re-invoked to resume EMM receiving.

Parameters:

CASModule casModule – Instance of CAS module.

Returns:

Success – CASModuleManager.ACTION_OK.

Failure – Returns the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS -Invalid parameters.
CASModuleManager.ACTION_ERROR_DRIVER -Driver error.

B.2.10 Class JSDCAS.CASPacketEvent

B.2.10.1 getCableModemFilter

Prototype:

{number|string}getCableModemFilter ( )

Description:

Returns the cableModemFilter used for filtering packet. Returns Null if Cable Modem DSG is not used.

Returns:

ADSG mode – returns CAS Tuner ID, numeric.

BDSG mode – Returns virtual MAC address.

B.2.10.2 getPacketData

Prototype:

{uint8Array}getPacketData ( )

Description:

Returns packet data.

B.2.10.3 getPacketHeader

Prototype:

{uint8Array}getPacketHeader ( )

Description:

Returns data packet header, which includes IP address and UDP header.

B.2.10.4 getSourceURL

Prototype:

{string}getSourceURL ( )

Description:

Returns the source URL for receiving CAS data packet by UDP.

Returns:

Source URL string

B.2.11 Class JSDCAS.CASSession

For every descrambling request, the platform generates a CAS Session object, which contains a unique session ID and all information about the playing of the program, as well as the CA descriptor in the PMT related to the descrambling. For each descrambling request, JS DCAS App uses CASModule.onStartDescrambling to obtain CAS Session. The CASSession object can also be used in receiving CAT update message, in which case only some fields of the CASSession object is valid.

B.2.11.1 GetCasDescriptor

Prototype:

{CASDescriptor}getCasDescriptor ( )

Description:

Returns CA descriptor, which can either be from PMT or CAT.

Returns:

Instance of CA descriptor

B.2.11.2 getChannelNumber

Prototype:

{number}getChannelNumber ( )

Description:

Returns channel number. This method is optional, especially for platforms without defined channel numbers (can return 0)

Returns:

Channel number

B.2.11.3 getNetworkId

Prototype:

{number}getNetworkId ( )

Description:

Returns original network ID. This method is optional. The platform can return 0 if unable to obtain it.

Returns:

original network ID

B.2.11.4 getOperationType

Prototype:

{number}GetOperationType ( )

Description:

Returns operation type.

Returns:

Value for operation type

CASSession.OPERATION_TYPE_PRESENTATION
CASSession.OPERATION_TYPE_RECORDING
CASSession.OPERATION_TYPE_BUFFERING

CASSession.OPERATION_TYPE_SECOND_DEVICE .

B.2.11.5 getProgramNumber

prototype:

{number}getProgramNumber ( )

Description:

Returns program number

B.2.11.6 getServiceIdentifier

Prototype:

{number|*}getServiceIdentifier ( )

Description:

Returns the identifier of the service being descrambled. This identifier can be a value or an object.

B.2.11.7 getSessionId

Prototype:

{number}getSessionId ( )

Description:

Returns Session ID

B.2.11.8 getStreamPath

Prototype:

{Uint8Array} getStreamPath ( )

Description:

Returns the StreamPath data

B.2.11.9 getStreamPIDs

Prototype:

{Array} getStreamPIDs ( )

Description:

Returns the Stream PIDs list.

B.2.11.10 getStreamTypes

Prototype:

{Array} getStreamTypes ( )

Description:

Returns the StreamTypes list. The StreamType is defined in [ISO/IEC 13818-1].

B.2.11.11 getTransmitterScramblingMode

Prototype:

{number}getTransmitterScramblingMode ( )

Description:

Returns value for descrambling mode

B.2.11.12 getTransportStreamId

Prototype:

{number}getTransportStreamId ( )

Description:

Returns TS ID of the service being descrambled

B.2.11.13 getTunerId

Prototype:

{number}getTunerId ( )

Description:

Returns the Tuner ID of the service being descrambled.

B.2.12 Class JSDCAS.CASStatus

JS DCAS App uses this object to report descrambling status to the platform. Every time the descrambling status changes, JS DCAS App should invoke CASModuleManager.sendDescramblingEvent to notify the platform. After receiving the status change, the platform can either handle by itself or forward it to UI application. UI application can notify user of descrambling success or failure simply by pop-up OSD, or display details about why descrambling fails, by analyzing additional information in the CASStatsus object. Format of additional information is project specific. If there is no additional information added by JS DCAS App, the UI application can also use the token obtained from the CASStatus object to obtain more information by communicating with JS DCAS App using inter-process communication (IPC) or other methods provided by the platform.

B.2.12.1 Status Value List

JSDCAS.CASStatus.CONTENT_PROBLEM_COMMUNICATION_ERROR

JSDCAS.CASStatus.CONTENT_PROBLEM_GENERAL_ERROR

JSDCAS.CASStatus.CONTENT_PROBLEM_HARDWARE_FAILURE_SOC

JSDCAS.CASStatus.CONTENT_PROBLEM_INVALID_CA_PACKET

JSDCAS.CASStatus.CONTENT_PROBLEM_MISSING_KEY

JSDCAS.CASStatus.CONTENT_PROBLEM_NO_CA_PACKET

JSDCAS.CASStatus.CONTENT_PROBLEM_NONE

JSDCAS.CASStatus.CONTENT_PROBLEM_POSITION_NOT_LEGAL_BLOCKING

JSDCAS.CASStatus.CONTENT_PROBLEM_POSITION_NOT_LEGAL_GRACE

JSDCAS.CASStatus.CONTENT_PROBLEM_POSITION_NOT_LEGAL_WARNING

JSDCAS.CASStatus.CONTENT_PROBLEM_POSITION_NOT_READY_BLOCKING

JSDCAS.CASStatus.CONTENT_PROBLEM_POSITION_NOT_READY_WARNING

JSDCAS.CASStatus.CONTENT_PROBLEM_PR_LIMIT_EXCEEDED

JSDCAS.CASStatus.CONTENT_PROBLEM_SERVICE_NOT_AUTHORIZED

JSDCAS.CASStatus.CONTENT_PROBLEM_SUBSCRIBER_NOT_AUTHORIZED

JSDCAS.CASStatus.CONTENT_PROBLEM_TRANSITION_WARNING

B.2.12.2 Methods

B.2.12.2.1 getCasToken

Prototype:

{number}getCasToken ( )

Description:

Returns CAS token. If the platform forwards information of CASStatus to UI application, UI application can use this token to request JS DCAS application for detailed status information. The request method such as IPC, is decided by the platform.

B.2.12.2.2 getMajorContentProblem

Prototype:

{number}getMajorContentProblem ( )

Description:

Returns a value representing the reason for not being able to view program.

B.2.12.2.3 getStatusData

Prototype:

{ArrayBuffer}getStatusData ( )

Description:

This method returns extended status data that the JSDCAS application attached to the CAS status object. With this additional status data, the UI application can show more detailed information regarding descrambling status to user.

Returns:

Returned data can be in the ArrayBuffer type. Shall return null if no extended data to provide.

B.2.12.2.4 isSuccess

Prototype:

{boolean}isSuccess ( )

Description:

Returns descrambling status, which can be either success or failure.

Returns:

True – Success. False – Failure.

B.2.13 Class JSDCAS.TeeController

Controller of communication between JS DCAS and TEE.

B.2.13.1 Methods

B.2.13.1.1 sendCommandToTEE

Prototype:

```
{TeeRetVal}sendCommandToTEE ( teeAppUUID,
                              commandId,
                              inputData,
                              applicationContext)
```

Description:

JS DCAS App uses this method to send command to TA running in TEE.

Parameters:

Uint8Array teeAppUUID – The universally unique identifier (UUID) of TA, 16 bytes. Every CA vendor has different ID.

number commandId – Command ID in TEE communication, defined by each CA vendor itself.

Uint8Array inputData – Data sent to TA

* applicationContext – Application context which is platform specific. Usually provided to application by the platform during initiation.

Returns:

Returns the TeeRetVal object. This object contains information returned from TA, such as data or error, etc.

B.2.14 Class JSDCAS.TeeRetVal

This object is returned by TeeController.sendCommandToTEE and contains information returned from TEE, such as data and error, etc.

B.2.14.1 Returned Value List

JSDCAS.TeeRetVal.TEEC_ERROR_ACCESS_CONFLICT

JSDCAS.TeeRetVal.TEEC_ERROR_ACCESS_DENIED

JSDCAS.TeeRetVal.TEEC_ERROR_BAD_FORMAT

JSDCAS.TeeRetVal.TEEC_ERROR_BAD_PARAMETERS

JSDCAS.TeeRetVal.TEEC_ERROR_BAD_STATE

JSDCAS.TeeRetVal.TEEC_ERROR_BUSY

JSDCAS.TeeRetVal.TEEC_ERROR_CANCEL

JSDCAS.TeeRetVal.TEEC_ERROR_COMMUNICATION

JSDCAS.TeeRetVal.TEEC_ERROR_EXCESS_DATA

JSDCAS.TeeRetVal.TEEC_ERROR_FSYNC_DATA

JSDCAS.TeeRetVal.TEEC_ERROR_GENERIC

JSDCAS.TeeRetVal.TEEC_ERROR_INVALID_CMD

JSDCAS.TeeRetVal.TEEC_ERROR_ITEM_NOT_FOUND

JSDCAS.TeeRetVal.TEEC_ERROR_MAC_INVALID

JSDCAS.TeeRetVal.TEEC_ERROR_NO_DATA

JSDCAS.TeeRetVal.TEEC_ERROR_NOT_IMPLEMENTED

JSDCAS.TeeRetVal.TEEC_ERROR_NOT_SUPPORTED

JSDCAS.TeeRetVal.TEEC_ERROR_OUT_OF_MEMORY

JSDCAS.TeeRetVal.TEEC_ERROR_READ_DATA

JSDCAS.TeeRetVal.TEEC_ERROR_REGISTER_EXIST_SERVICE

JSDCAS.TeeRetVal.TEEC_ERROR_RENAME_OBJECT

JSDCAS.TeeRetVal.TEEC_ERROR_SECURITY

JSDCAS.TeeRetVal.TEEC_ERROR_SEEK_DATA

JSDCAS.TeeRetVal.TEEC_ERROR_SERVICE_NOT_EXIST

JSDCAS.TeeRetVal.TEEC_ERROR_SESSION_MAXIMUM

JSDCAS.TeeRetVal.TEEC_ERROR_SESSION_NOT_EXIST

JSDCAS.TeeRetVal.TEEC_ERROR_SHORT_BUFFER

JSDCAS.TeeRetVal.TEEC_ERROR_TAGET_DEAD_FATAL

JSDCAS.TeeRetVal.TEEC_ERROR_TRUNCATE_OBJECT

JSDCAS.TeeRetVal.TEEC_ERROR_TRUSTED_APP_LOAD_ERROR

JSDCAS.TeeRetVal.TEEC_ERROR_WRITE_DATA

JSDCAS.TeeRetVal.TEEC_ORIGIN_API

JSDCAS.TeeRetVal.TEEC_ORIGIN_COMMS

JSDCAS.TeeRetVal.TEEC_ORIGIN_JS_LAYER

JSDCAS.TeeRetVal.TEEC_ORIGIN_NOT_SPECIFIED

JSDCAS.TeeRetVal.TEEC_ORIGIN_TEE

JSDCAS.TeeRetVal.TEEC_ORIGIN_TRUSTED_APP

JSDCAS.TeeRetVal.TEEC_SUCCESS

B.2.14.2 Method

B.2.14.2.1 getOriginCode

prototype:

{number}getOriginCode ( )

Description:

Returns origin code.

B.2.14.2.2 getResponseData

prototype:

{Uint8Array}getResponseData ( )

Description:

To get data returned from TA

Returns:

Data returned from TA, which can be Null for some commands. Returns Null if error occurs in invocation or communication.

B.2.14.2.3 getReturnCode

Prototype:

{number}getReturnCode ( )

Description:

Returns to retrieve return code.

B.3 Other GP extension APIs

B.3.1 Cryptography and signature verification APIs

B.3.1.1 Data types and structures

B.3.1.1.1 Basic data types

typedef unsigned int TEE_Result;

typedef unsigned int uint32_t;

B.3.1.1.2 Enums Returned

#define TEE_SUCCESS 0;

B.3.1.2 APIs definitions

B.3.1.2.1 TEE_SM2_Verify

To verify SM2 signature.

Prototype:

```
TEE_Result  TEE_SM2_Verify (unsigned char *pub_key,
                            size_t pub_key_len,
                            unsigned char *hash,
                            size_t hash_len,
                            unsigned char *sig,
                            size_t sig_len)
```

Inputs:

pub_key: SM2 Public key consisting of 1 byte header and 64-byte public key data, totally 65 bytes

pub_key_len: SM2 Public key size

hash: SM3 hash value of the content, 32 bytes

hash_len: Hash value size

sig:Signature, 64 bytes

sig_len: Length of signature

Outputs:

N/A

Returns:

TEE_SUCCESS: Signature verification success

Other values: signature verification failure

B.3.1.2.2 TEE_Perform_SM3

To compute SM3 hash.

Prototype:

```
TEE_Result   TEE_Perform_SM3 (  unsigned char* dataIn,
                                unsigned int dataInLen,
                                unsigned char* result);
```

Inputs:

dataIn: content data over which the SM3 hash is to be computed

dataInLen: content size

Outputs:

result: hash value retrieved, 32 bytes

Returns:

TEE_SUCCESS:computation successful

Other values: computation failure

B.3.1.2.3 TEE_SM2_Encrypt

To encrypt data with SM2 algorithm and public key.

Prototype:

```
TEE_Result  TEE_SM2_Encrypt ( unsigned char *pub_key,
                              size_t pub_key_len,
                              uint8_t *inputData,
                              uint32_t inputData_size
                              uint8_t *outputData,
                              uint32_t outputData_size)
```

Inputs:

pub_key: SM2 Public key, consisting of a 1-byte header and 64-byte public key data, totally 65 bytes

pub_key_len: Length of SM2 public key

inputData: input data to be encrypted

inputData_size: size of the input data

outputData:data buffer to store the encrypted data

outputData_size: data buffer to store the size of the encrypted data, it should be the input data size plus 96 bytes

Outputs:

outputData: encrypted data stored in the outputData buffer, it is a concatenation of C1|C3|C2, where C1 is the EC point randomly generated, C2 is the encrypted message, C3 is the hash value related to C1 and C2.

Returns:

TEE_SUCCESS: Encryption success

Other values: Encryption failure

B.3.1.2.4 TEE_Perform_CRC

To compute the cyclic redundancy check (CRC) value.

Prototype:

```
TEE_Result   TEE_Perform_CRC (  int mode,
                                unsigned char* dataIn,
                                unsigned int dataInLen,
                                unsigned char* result );
```

Inputs:

mode: CRC computing mode, 0=CRC16, 1=CRC

dataIn: the data over which the CRC will be computed

dataInLen: Length of input data

Outputs:

result: the CRC result computed, it is of 2 bytes for CRC16, 4 bytes for CRC32

Returns:

TEE_SUCCESS: CRC computation success

Other values: computation failure

B.3.1.2.5 TEE_GenerateRandom

To generate a random number.

Prototype:

```
TEE_Result TEE_GenerateRandom (  void* randomBuffer,
                                 size_t randomBufferLen );
```

Inputs:

randomBuffer: data buffer for storing the random number

randomBufferLen: data buffer length

Outputs:

randomBuffer: the data buffer with a random number stored

Returns:

TEE_SUCCESS: Generation success

Others: Generation failure

B.3.1.2.6 TEE_SM4_Encrypt

To encrypt data with SM4 algorithm.

Prototype:

```
TEE_Result    TEE_SM4_Encrypt (  int mode,
                                 uint8_t *IV,
                                 uint8_t *key,
                                 uint8_t *inputData,
                                 uint8_t *outputData,
                                 uint32_t data_size);
```

Inputs:

mode: encryption mode, 0=ECB, 1=CBC

IV: Initialization vector. It is a 16-byte data in CBC mode, and should be ignored in ECB mode

key: SM4 key, 16 bytes

inputData: content to be encrypted

outputData: data buffer for storing the encrypted output

data_size: data size for both input and output data, it should be any multiple of 16 byptes, otherwise the encryption will fail

Outputs:

outputData: Data encrypted

Returns:

TEE_SUCCESS:encryption success

Others: failure

B.3.1.2.7 TEE_SM4_Decrypt

To decrypt data with SM4 algorithm

Prototype:

```
TEE_Result    TEE_SM4_Decrypt ( int mode,
                                uint8_t *IV,
                                uint8_t *key,
                                uint8_t *inputData,
                                uint8_t *outputData,
                                uint32_t data_size );
```

Inputs:

mode: decryption mode, 0=ECB, 1=CBC

IV: Initialization vector. It is a 16-byte data in CBC mode, and should be ignored in ECB mode.

key: SM4 key, 16 bytes

inputData: content to be decrypted

outputData: data buffer to store the decrypted output

data_size: data size for both input and output data. It should be multiple of 16 bytes, otherwise the decryption will fail.

Outputs:

outputData: data decrypted

Returns:

TEE_SUCCESS: Decryption success

Others: failure

B.3.2 Memory management APIs

B.3.2.1 Data types and structures

B.3.2.1.1 Basic data types

N/A

B.3.2.1.2 Enums returned

N/A

B.3.2.2 API definitions

B.3.2.2.1 TEE_MemFill

Fill a memory space with a specified value.

Prototype:

```
void TEE_MemFill ( void *buffer,
                   uint32_t x,
                   uint32_t size);
```

Inputs:

buffer: the starting address of the memory space to be filled

x: specified value for the filling

size: size of the memory space to be filled

Outputs:

N/A

Returns:

N/A

B.3.2.2.2 TEE_MemMove

Move data from one place to another in memory.

Protoytpe:

```
void TEE_MemMove (void *dest,
                  void *src,
                  uint32_t size);
```

Inputs:

dest: the starting address of the destination memory space

src: the starting address of the source memory space

size: size of the data to be moved

Outputs:

N/A

Returns:

N/A

B.3.3 Miscellaneous APIs

B.3.3.1 Data types and structures

B.3.3.1.1 Basic data types

N/A

B.3.3.1.2 Enums returned

N/A

B.3.3.2 API definitions

B.3.3.2.1 TEE_Printf_Func

Print logs.

Prototype:

```
void TEE_Printf_Func(const char * fmt, …);
```

Inputs:

fmt: format list for the printing

Outputs:

N/A

Returns:

N/A

B.4 Security chipset key ladder driver APIs

B.4.1 B.3 data types and structures

B.4.1.1 Basic data types

```
typedef unsigned char      TEE_KLAD_BYTE;
typedef unsigned short     TEE_KLAD_USHORT16;
typedef unsigned long        TEE_KLAD_ULONG32;
typedef unsigned char        TEE_KLAD_BOOLEAN;
```

B.4.1.2 Enums returned

```
typedef enum
{
    TEE_KLAD_OK,
    TEE_KLAD_FAIL,
    TEE_KLAD_UNMATCH_CHAN,
}TEE_KLAD_STATUS
```

B.4.2 API definitions

B.4.2.1 TEE_KLAD_Init

Initialize Key Ladder.

Prototype:

TEE_KLAD_STATUS TEE_KLAD_Init(void);

Inputs:

N/A;

Outputs:

N/A.

B.4.2.2 TEE_KLAD_Delnit

Deinitialize Key Ladder.

Prototype:

TEE_KLAD_STATUS TEE_KLAD_Delnit(void);

Inputs:

N/A;

Outputs:

N/A.

B.4.2.3 TEE_KLAD_GetChipId

Read security chipset's ChipId.

Prototype:

TEE_KLAD_STATUS TEE_KLAD_GetChipId(TEE_KLAD_BYTE* chipid);

Inputs:

N/A;

Outputs:

Security chipset's ChipId, 8 bytes buffer, allocated and freed by the application which calls this interface.

B.4.2.4 TEE_KLAD_GetResponseToChallenge

Compute the response according to the challenge.

Prototype:

```
TEE_KLAD_STATUS TEE_KLAD_GetResponseToChallenge
(
    TEE_KLAD_BYTE   *Nonce,
    TEE_KLAD_BYTE   NonceLength,
    int         keyDescriptorsLength,
    TEE_KLAD_BYTE   *keyDescriptors,
    TEE_KLAD_BYTE   *response,
    TEE_KLAD_BYTE   *responseLength
);
```

Inputs:
Nonce: challenge data
NonceLength: length of challenge data
keyDescriptorsLength: length of key descriptors
keyDescriptors: key descriptors
Outputs:
response: the computing result of challenge
responseLength: the length of the result
The descriptors in key descriptors is as following:
Key ladder descriptor's definition in byte:
0: ENCRYPTION_KEY_DSCR_TAG = 0x03
1: descriptor's length
2: level of key ladder, for challenge-response computing, set 2
3: length of the key used by key ladder
4-n: encrypted keys for key ladder
Algorithm descriptor's definition in byte:
0: ENCRYPTION_SCHEME_DSCR_TAG = 0x04
1: descriptor's length
2-3: enums value of algorithm: 0=TDES, 1=AES, 2=SM4
CA vendor ID's descriptor's definition in byte:
0: VENDOR_ID_DSCR_TAG = 0x05
1: descriptor's length = 2
2-3: CA vendor ID

B.4.2.5 TEE_KLAD_SetDescrambler

Set descrambling parameters and keys to invoke descrambling.
Prototype:

```
TEE_KLAD_STATUS TEE_KLAD_SetDescrambler
(
    int             streamPathLength,
    TEE_KLAD_BYTE   *streamPath;
    int             numberOfStreamPids,
    TEE_KLAD_BYTE   *streamPids,
    Int             OddkeyDescriptorsLength,
    TEE_KLAD_BYTE   *OddkeyDescriptor,
    Int             EvenkeyDescriptorLength,
    TEE_KLAD_BYTE   *EvenkeyDescriptor
);
```

Inputs:
streamPathLength: length of stream path for descrambling
streamPath: stream path for descrambling
numberOfStreamPids: pid numbers of descrambling stream program
streamPids: pids of descrambling stream program
OddkeyDescriptorsLength: length of odd key descriptor
OddkeyDescriptor: odd key descriptor
EvenkeyDescriptorLength: length of even key descriptor
EvenkeyDescriptor: even key descriptor
The descriptor in odd key descriptor and even key descriptor is as following:
Clear CW descriptor's definition in byte:
0: CLEAR_CW_DSCR_TAG = 0x01
1: descriptor's length
2-n: clear CW;

Encrypted CW descriptor's definition in byte:
0: ENCRYPTED_CW_DSCR_TAG = 0x02
1: descriptor's length
2-n: encrypted CW. To decrypt to get CW, additional descriptors will be provided.
Key ladder descriptor's definition in byte:
0: ENCRYPTED_KEY_DSCR_TAG = 0x03
1: descriptor's length
2: key level, 0 for CW, 1, 2… for other keys
3: key length
4-n: encrypted key
Key encryption algorithm descriptor's definition in byte:
0: ENCRYPTION_SCHEME_DSCR_TAG = 0x04
1: descriptor's length
2-3: enums value of algorithm: 0=TDES, 1=AES, 2=SM4
CA vendor ID's descriptor's definition in byte:
0: VENDOR_ID_DSCR_TAG = 0x05
1: descriptor's length = 2
2-3: CA vendor ID
Descrambling algorithm descriptor's definition in byte:
0: DESCRAMBLING_ALGORITHM_DSCR_TAG = 0x07
1: descriptor's length
2-3: enums value of algorithm: 0=DVB-CSA2[ETSI ETR 289], 1=CSA3[b-CSA3]
Outputs:
N/A

B.4.2.6 TEE_KLAD_StopDescrambler

Stop descrambling.
Prototype:

```
TEE_KLAD_STATUS TEE_KLAD_StopDescrambler
(
      int            streamPathLength,
      TEE_KLAD_BYTE  *streamPath;
      int            numberOfStreamPids,
      TEE_KLAD_BYTE  *streamPids,
);
```

Inputs:
streamPathLength: length of stream path for descrambling
streamPath: stream path for descrambling
numberOfStreamPids: pid numbers of descrambling stream program
streamPids: pids of descrambling stream program
Outputs:
N/A

Bibliography

[b-ITU-T J.93]	Recommendation ITU-T J.93 (1998), *Requirements for conditional access in the secondary distribution of digital television on cable television systems.*
[b-ITU-T J.290]	Recommendation ITU-T J.290 (2006), *Next generation set-top box core architecture.*
[b-ITU-T J.1026]	Recommendation ITU-T J.1026 (2019), *Downloadable conditional access system for unidirectional networks – Requirements*
[b-CSA3]	DVB-CSA – DVB BlueBook A125 (2008), *(Document a125_CSA3_dTR101289.v1.2.1), Support for use of the DVB Scrambling Algorithm version 3 within digital broadcasting systems.*
[b-ETSI TS 103 162]	ETSI TS 103 162 V1.1.1 (2010), A*ccess, Terminals, Transmission and Multiplexing (ATTM); Integrated Broadband Cable and Television Networks; K-LAD Functional Specification.*
[b-GB/T 32907]	GB/T 32907 (2016), *Information security technology – SM4 block cipher algorithm.*
[b-GY/T 255-2012]	GY/T 255-2012, *Technical specification of downloadable conditional access system*.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	Tariff and accounting principles and international telecommunication/ICT economic and policy issues
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	**Cable networks and transmission of television, sound programme and other multimedia signals**
Series K	Protection against interference
Series L	Environment and ICTs, climate change, e-waste, energy efficiency; construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling, and associated measurements and tests
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	Global information infrastructure, Internet protocol aspects, next-generation networks, Internet of Things and smart cities
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2020