Application Programming Interface Specification V1.0

for

Automatic Vending & Recharge Machine

	CRIS				
	Name	Date	Signature	Name	Signature
Prepared and revised by					
Reviewed and accepted by					

Revision History

Release No.	Name	Date	Revision Description/Reason for Changes
Rev. 1.0	CRIS	01/08/2019	Initial Work

TABLE OF CONTENTS

1.0

GENERAL INFORMATION

1. GENERAL INFORMATION:

1.1. Introduction:

The document is intended to introduce the functional and technical requirements of the AVRM device APIs to be implemented by the developer of the APIs. Suppliers of the AVRM Kiosk will have to deliver the APIs strictly following the API design and specifications written in this document in order to enable the application programmers of the AVRM Software to interface with the AVRM devices.

1.2. System Overview:

As per the functional and technical requirements of the AVRM Kiosk, it is integrated with a number of electro mechanical and electronic devices for automation of Smart Token/Card issue and Smart Card recharge operations of Metro Railway, Kolkata. Commuters of the Metro will be able to buy Single Journey Tokens, new multi-ride Stored Value Smart Cards and recharge Stored Value Smart Cards by paying currency Coins and Notes at their own through the self service Kiosks. AVRM Kiosk software is a java based application software for monitoring/controlling the devices to achieve the AVRM functionalities which requires java based interfaces to the connected devices.

This API specification document will discuss in detail how the APIs need to be implemented for the different devices of the AVRM.

1.3. Organization of the document:

This document is divided into 4 major sections listed and defined below:

- General Information
- API Architecture and Configuration
- API Structure
- API Specifications

1.4. ACRONYMS AND ABBREVIASIONS:

1D	1 (One) Dimentional
2D	2 (Two) Dimentional
AFC	Automatic Fare Collection
AN1	ANti collision loop 1
AN2	ANti collision loop 2
AN1 & SELECT	ANti collision and SELECTion loop 1
AN2 & SELECT	ANti collision and SELECTion loop 2
APDU	Application Protocol Data Unit
API	Application Programming Interface
ATQA	Answer To reQuest, type A
ATR	Answer To Reset
AVRM	Automatic Vending and Recharge Machine
Cris	Centre for railway information systems
CSC	Contactless Smart Card
CST	Contactless Smart Token
DESFire	Data Encryption Standard based MIFARE Contact less smart card
DESFire Native APDU	Non-ISO proprietary commands for DESFire
EV1	Evaluation Version 1
FW	Firmware
H/W	Hardware
IEC	International Electrotechnical Commission
INR	INdian Rupee
ISO	International Organization for Standardization
ISO 14443A	Type-A Contact less smart card standard
ISO 14443A-3	Part-3 of ISO 14443A
ISO 14443A-4	Part-4 of ISO 14443A
ISO 7816	ISO-7816 Contact Card Standard
ISO 7816-4	Part-4 of ISO 7816
ISO/IEC 7816 wrapped APDU	ISO-14443A-4 APDU wrapped into ISO-7816 frame
MIFARE	NXP's Contact less smart card product name

MIFARE DESFire	0.6 Version of DESFire product
MIFARE DESFire EV1	Evaluation Version 1 of DESFire product
MIFARE SAM AV1	MIFARE SAM Version 1
MIFARE SAM AV2	MIFARE SAM Version 2
MIFARE Ultralight	Contact less memory card of MIFARE product family
OS	Operating System
png	Portable Network Graphics, an image format
PPS	Protocol and Parameter Selection
RATS	Request for Answer To Select
REQA	REQuest Command, type A
RF	Radio Frequency
RFID	Radio Frequency Identification
RFU	Reserved for Future Use
R/W	Reader/Writer
RS232	Recommend Standard number 232
S/W	Software
SAM	Secure Application Module
T=1	ISO-7816 Contact Card transmission protocol
T=CL	ISO-14443A Contact less Card transmission protocol
TCU	Token Capture Unit
UID	Unique IDentifier
Ultralight	Contact less memory card of MIFARE product family
UPS	Un-interrupted Power Supply
USB	Universal Serial Bus
-ve	Negative

2.0

API ARCHITECTURE AND CONFIGURATION

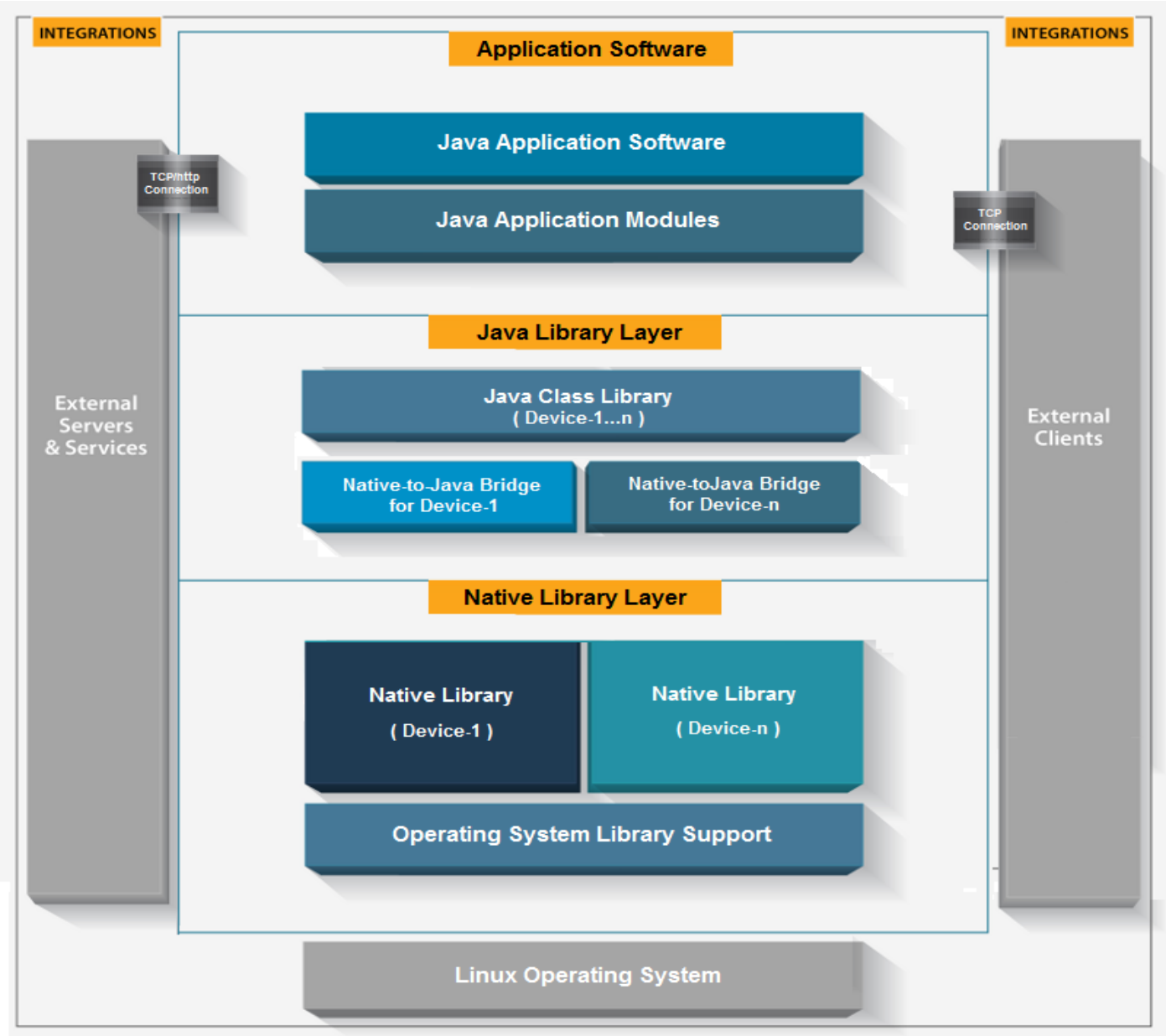
2. API ARCHITECTURE AND CONFIGURATION:

Purpose of this section is to describe the communication architecture and the configuration of the AVRM APIs.

Note

2.1. Architecture

The AVRM device APIs need to be a java based class library in order to enable direct integration with the Java based AVRM Application Software. However, relevant part of the library functionalities can be implemented using native code to achieve the required performance criteria of the device functionalities wherever applicable. A reference model of such hybrid implementation of device APIs is as indicated in the following schematic diagram:



2.2. Configuration

2.2.1. API Access Configuration

Device wise configuration information can be written in a single AVRM.cfg configuration file in the following format:

```
[DEVICE NAME1]
PARAMETER1=VALUE
PARAMETER2=VALUE
PARAMETER3=VALUE
..
PARAMETERn=VALUE

[DEVICE NAME2]
PARAMETER1=VALUE
PARAMETER2=VALUE
PARAMETER3=VALUE
..
PARAMETERn=VALUE

.
.
.
[DEVICE NAMEn]
PARAMETER1=VALUE
PARAMETER2=VALUE
PARAMETER3=VALUE
..
PARAMETERn=VALUE
```

If this configuration file contains a parameter which is also being passed by any individual API method; the parameter passed to the API method will take precedence over the corresponding parameter in the configuration file.

2.2.2. API Logging Configuration

The AVRM device interfacing APIs should write different levels of informational, warning and error events in a file while the APIs are invoked throughout the life cycle of the AVRM application software to help troubleshoot the different error conditions that may arise. It is recommended to write different levels of events in a concise way to ensure the required throughput of the APIs.

List of logging levels the API should provide:

TRACE (41): This level specifies finer-grained informational messages than the DEBUG.
DEBUG (42): This level specifies fine-grained informational messages that are most useful to debug an application.
INFO (43): This level specifies informational messages that highlight the progress of the application at coarse-grained level.
WARN (44): This level specifies potentially harmful situations.
ERROR (45): This level specifies error messages that might still allow the application to continue running.
FATAL (46): This level specifies very severe error messages that will presumably lead the application to abort.
ALL (47): The ALL has the lowest possible rank and is intended to turn on all logging.
OFF (40): The OFF has the highest possible rank and is intended to turn off logging.

While ALL and OFF are special logging levels and should be used in extreme situations, TRACE is of the lowest priority and FATAL is having highest priority.


Note

Logging information require to be written in a day wise log file pattern:
"AVRMAPI_<Host Name>Log<YYYY>.<Current day of the year> where

<Host Name>: is the host name of the machine.
<YYYY>: is the 4 digit current year

<Current day of the year>: 1 is the first day of the year e.g. 1st January, 31 is the 31st day of the year e.g. 31st January.

In **LOGLEVEL (x), x** is the numeric value of the logging level corresponding to the LOGLEVEL.

Irrespective of the LOGLEVEL, any relevant event to be logged with timestamp upto milliseconds precision LOGLEVEL and Device Id/Name at the begining of the message. As for example a message at DEBUG level should look like below:
[11:25:35.268 DEBUG <Device Id/Name>] .....


3.0
3.0
API STRUCTURE

3. API STRUCTURE:

The sole purpose of this section is to describe the structure of the java class library for different devices of the AVRM Kiosk. The AVRM device APIs (java class library) are a set of java classes packaged in jar archive. Dependent Native libraries working under the java class library requires to be packaged in the same jar archive and should be loaded from the archive during run time. Java class library, package and class names are as detailed below:

3.1. Java Class Library Name (AVRM API)

AVRMApi.jar

3.2. Java Package Name

Cris

3.3. Java Class Names for device interfaces

3.3.1. Common.class

This class implements the non-functional aspects of the API like, Vendor Information, Java Class Library version etc.

3.3.2. SmartCard.class

This class implements the functionalities of the Smart Card Accepter/Dispenser and integrated Contact and Contactless (RFID) R/Ws.

3.3.3. Currency.class

This class implements the functionalities of the Currency Note/Coin Accepter and escrows.

3.3.4. TokenDispenser.class

This class implements the functionalities of the Token Dispenser and integrated Contact and Contactless (RFID) R/Ws.

3.3.5. Security.class

This class implements the status and functionalities of the Security Doors of the Kiosk.

3.3.6. UPS.class

This class implements the status and functionalities of the UPS.

3.3.7. Printer.class

This class implements the status and functionalities of the Printer.

4.0

API SPECIFICATIONS

4. API SPECIFICATIONS:

Purpose of this chapter is to describe the java methods corresponding to the functionalities of each AVRM device in terms of signature, input parameters and return values. Each java class corresponding to a device or group of devices has the following java methods for:

- Connecting the device
- Getting version information
- Getting device status
- Disconnecting the device and
- Executing relevant device functionalities

Note

If any java class does not have dependency on a native library then "Get Native Lib Version" methods specified in the subsequent sections should return corresponding Library Versions implemented in Java.

4.1. COMMON API SPECIFICATION

The non-functional aspects of the API like, Vendor Information, Java Class Library version etc. are included in this section of API specification.

4.1.1. Get Vendor Id

Purpose	To get the vendor id of the vendor supplying the API		
Package Name	Cris	**Class Name**	Common
Method Signature	Int GetVendorId()		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Return Values : Vendor Id of the vendor			
Value	**Description**		
0	Error retrieving vendor id		
1	Mega Designs Pvt. Ltd.		
2	Lipi Data Systems Ltd.		

4.1.2. Get AVRMApi Version

Purpose	To get the AVRMApi Version		
Package Name	Cris	**Class Name**	Common
Method Signature	String GetAVRMApiVersion()		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Return Values : Version information comprising the following: **MAJOR -** Incompatible API changes **MINOR -** Functionality adding in a backwards-compatible manner **BUILD -** Bug fixing backwards-compatible			
Value	**Description**		
00.00.00	Error retrieving version information		
01.00.00	A valid version string		

4.1.3. Set Logging Level

Purpose	To activate required logging level		
Package Name	Cris	**Class Name**	Common
Method Signature	int SetLoggingLevel(int LogLevel)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
LogLevel	Level of logging to be enabled. Refer "2.2.2 API Logging Configuration" for the value of the level.		
Return Values :			
Value	**Description**		
0	Log level set successfully		
1	Setting log level failed		

4.1.4. Get Logging Level

Purpose	To get the currently activated logging level		
Package Name	Cris	**Class Name**	Common
Method Signature	int GetLoggingLevel()		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Return Values : Enabled level of logging. Refer "2.2.2 API Logging Configuration" for the value of the level.			
Value	**Description**		
x	Currently activated Log level		
1	Getting log level failed		

4.2. CARD ACCEPTER/DISPENSER DEVICE & CONTACT/CONTACTLESS R/W API SPECIFICATION

The methods implementing the functionalities of the Smart Card Accepter/Dispenser and integrated Contact and Contactless (RFID) R/Ws are listed in this section.

4.2.1.CARD ACCEPTER/DISPENSER DEVICE API

Smart Card Accepter/Dispenser functionality related methods are included in this sub-section.

4.2.1.1. Connect Device

Purpose	To connect to the Card Accepter/Dispenser & RFID Reader combo device		
Package Name	Cris	**Class Name**	SmartCard
Method Signature	int ConnectDevice(int PortId, int ChannelClearanceMode, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
PortId	Serial Port Number to which the device is connected	0-50	If the device uses a serial port.
		51	In case the device uses USB port
ChannelClearanceMode	If there is a card in the device channel either it will be sent to rejection bin or return from the mouth of the device or kept in its position.	0	Retain in the channel
		1	Send to rejection bin
		2	Return from the mouth of the device
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values			
Value	**Description**		
0	Device connected successfully		
1	Channel clearance failed due to rejection bin full		
2	Channel clearance failed due to return mouth blocked		
3	Channel clearance failed due to unknown reason		
18	Operation timeout occurred		
20	Device already connected		
25	Port doesn't exist		
26	Port doesn't exist		
27	Port doesn't exist		
28	Communication failure		
29	Communication failure		
30	Communication failure		
31	Other error		

Note

In case the combo device uses separate ports for Card Accepter/Dispenser and RFID Reader, RFID Reader needs to be connected using the ConnectDevice method described under section 4.2.2.1

4.2.1.2. Get Native Lib Version

Purpose	To get the Native Library Version		
Package Name	Cris	**Class Name**	SmartCard
Method Signature	String GetSCardDevNativeLibVersion()		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Return Values : Version information comprising the following: **MAJOR -** Incompatible API changes **MINOR -** Functionality adding in a backwards-compatible manner **BUILD -** Bug fixing backwards-compatible			
Value	**Description**		
00.00.00	Error retreiving version information		
01.00.00	A valid version string		

4.2.1.3. Get Smart Card Device Firmware Version

Purpose	To get the Smart Card Device FW Version		
Package Name	Cris	**Class Name**	SmartCard
Method Signature	String GetSCardDevFWVersion()		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Return Values : Version information			
Value	**Description**		
00.00.00	Error retreiving version information		
01.00.00	A valid version string in the format or equivalent as received by the device		

4.2.1.4. Get Smart Card Reader Firmware Version

Purpose	To get the Reader FW Version		
Package Name	Cris	**Class Name**	SmartCard
Method Signature	String GetSCardReaderFWVersion()		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**

Return Values : Version information	
Value	**Description**
00.00.00	Error retreiving version information
01.00.00	A valid version string in the format or equivalent as received by the device

This method should be implemented only if Smart Card Reader is accessed through the same communication port as Smart Card Accepter/Dispenser device.

4.2.1.5. Device Status

Purpose	To collect status of different components of the device		
Package Name	Cris	**Class Name**	SmartCard
Method Signature	byte[] GetDeviceStatus(int ComponentId, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
ComponentId	Identifier of the device sub-components	0	All component
		1	Reader
		2	Stacker
		3	Rejection Bin
		4	Channel
		5	Collection Bin
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values : Execution status of the API and status of the device components to be returned as a byte array as defined below:			
Value	**Description**		
Byte0	**Execution status of the API**		
0	Operation successful		
18	Operation timeout occurred		
28	Communication failure (Main Module)		
29	Communication failure (Reader)		
31	Other error		
Byte1	**RFID Reader status**		
0	Ready		
1	Not Ready		
Byte2	**SAM Reader status**		
0	Ready		

1	Not Ready
Byte3	**Stacker Status**
0	Empty
1	Nearly empty
2	Nearly full
3	Full
Byte4	**Approx. card count in stacker**
Byte5	**Rejection Bin status**
0	Empty
1	Nearly empty
2	Nearly full
3	Full
Byte6	**Card count in Rejection Bin**
Byte7	**Channel status**
0	Clear
1	Blocked
Byte8	**Channel sensor status** **8 bits where each bit will indicate whether a sensor is blocked or not; If no of sensors is less than 8 higher significant bits will be filled with 0's.**
0	Clear
1	Blocked
Byte9	**Collection Bin status**
0	Empty
1	Nearly empty
2	Nearly full
3	Full
Byte10	**Card count in Collection Bin**

Note

When ComponentId is other than 0, API will fill relevant values for the requested component leaving other component status 0; however execution status of the API is mandatory for any value of ComponentId.

Note

This API will be invoked frequently even when the devices are busy in accepting/dispencing smart cards during the life cycle of calling application.

4.2.1.6. Enable Card Acceptance

Purpose	It will enable the device to accept a card from front side while insertion slot or channel is not blocked. The API should return immediately after enabling card acceptance. After enabling card acceptance when a card is inserted, device should transport it to a position in the channel where antenna of the RFID Reader is placed for processing the card.		
Package Name	Cris	**Class Name**	SmartCard
Method Signature	int EnableCardAcceptance(int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values			
Value	**Description**		
0	Operation successful		
1	Channel blocked		
2	Insertion/return mouth blocked		
18	Operation timeout occurred		
20	Device not yet connected (Main Module)		
28	Communication failure		
31	Other error		

4.2.1.7. Disable Card Acceptance

Purpose	It will disable card acceptance from front side. The API should return immediately after disabling card acceptance. After disabling card acceptance when a card is inserted device should prohibit it by closing the insertion slot of the device.		
Package Name	Cris	**Class Name**	SmartCard
Method Signature	int DisableCardAcceptance(int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values			
Value	**Description**		
0	Operation successful		
18	Operation timeout occurred		
20	Device not yet connected (Main Module)		
28	Communication failure (Main Module)		

31	Other error

4.2.1.8. Accept Card

Purpose	It will enable the device to accept a card from front side while insertion slot or channel is not blocked. After enabling card acceptance the API should wait for insertion of a card for a predefined time period; if it finds a card at insertion slot the device should transport it to a position in the channel where antenna of the RFID Reader is placed for processing the card and thereafter the API will return. If within the predefined time period the API doesn't find any card or any blocking in the channel is detected, it will return with relevant error code as specified in return value of the API.		
Package Name	Cris	**Class Name**	SmartCard
Method Signature	int AcceptCard(int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values			
Value	**Description**		
0	Operation successful		
1	Channel blocked		
2	Insertion/return mouth blocked		
18	Operation timeout occurred		
20	Device not yet connected (Main Module)		
28	Communication failure		
31	Other error		

4.2.1.9. Dispense Card

Purpose	It dispenses a card from the stacker and position it in the channel where antenna of the RFID Reader is placed for processing the card and thereafter the API will return. If within the predefined time period the API doesn't find any card in the stacker or any blocking in the channel is detected, it will return with relevant error code as specified in return value of the API.		
Package Name	Cris	**Class Name**	SmartCard
Method Signature	int DispenseCard(int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values			
Value	**Description**		

0	Operation successful
1	Channel blocked
2	Insertion/return mouth blocked
3	Stacker empty
18	Operation timeout occurred
20	Device not yet connected (Main Module)
28	Communication failure
31	Other error

4.2.1.10. Return Card

Purpose	It returns a card from the escrow/reader position and hold it at the return mouth of the device until it is taken out by the customer or dispense it immediately; thereafter the API will return. If within the predefined time the API doesn't find any card in the channel or any blocking in the channel is detected, it will return with relevant error code as specified in return value of the API.		
Package Name	Cris	**Class Name**	SmartCard
Method Signature	int ReturnCard(intDispenseMode, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
DispenseMode	This mode will indicate whether the device will hold the card at the return mouth of the device or directly dispense it immediately.	0	Hold at the mouth of the device until it is taken out by the customer
		1	Dispense it immediately
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		

Return Values	
Value	**Description**
0	Operation successful
1	Return mouth blocked
2	No card in the channel
18	Operation timeout occurred
20	Device not yet connected (Main Module)
28	Communication failure
31	Other error

4.2.1.11. Reject Card

Purpose	It transports a card from the escrow/reader position to rejection bin of the device if the rejection bin is not full and thereafter the API will return. If within the predefined time period specified with Timeout parameter, the API doesn't find any card in the channel or any blocking in the channel is detected, it will return with relevant error code as specified in return value of the API.		
Package Name	Cris	**Class Name**	SmartCard
Method Signature	int RejectCard(int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values			
Value	**Description**		
0	Operation successful		
1	Rejection bin full		
2	No card in the channel		
18	Operation timeout occurred		
20	Device not yet connected (Main Module)		
28	Communication failure		
31	Other error		

4.2.1.12. Collect Card

Purpose	It transports a card from the escrow/reader position to collection bin of the device if the collection bin is not full and thereafter the API will return. If within the predefined time period specified with Timeout parameter, the API doesn't find any card in the channel or any blocking in the channel is detected, it will return with relevant error code as specified in return value of the API.		
Package Name	Cris	**Class Name**	SmartCard
Method Signature	int CollectCard(int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values			
Value	**Description**		
0	Operation successful		
1	Collection bin full		
2	No card in the channel		
18	Operation timeout occurred		

20	Device not yet connected (Main Module)
28	Communication failure
31	Other error

4.2.1.13. Is Card in the Channel

Purpose	This API can be used at any time after the connection is made; especially after EnableCardAcceptance or DispenseCard API is called to determine whether a card is reached at the reader position for processing.		
Package Name	Cris	**Class Name**	SmartCard
Method Signature	int IsCardInChannel(int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values			
Value	**Description**		
0	No card in the channel		
1	Card found in the channel		
18	Operation timeout occurred		
20	Device not yet connected (Main Module)		
28	Communication failure		
31	Other error		

.

4.2.1.14. Is Card Removed

Purpose	This API can be used after a card is returned and held at return mouth of the device to determine whether the returned card has been taken out by the customer.		
Package Name	Cris	**Class Name**	SmartCard
Method Signature	int IsCardRemoved(int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values			
Value	**Description**		
0	Not removed		
1	Removed		
18	Operation timeout occurred		
20	Device not yet connected (Main Module)		

28	Communication failure
31	Other error

4.2.1.15. Disconnect Device

Purpose	This API will be used to disconnect the device.		
Package Name	Cris	**Class Name**	SmartCard
Method Signature	int DisConnectDevice(int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values			
Value	**Description**		
0	Disconnected successfully		
1	Disconnected successfully but a card is in the channel		
18	Operation timeout occurred		
20	Device not yet connected (Main Module)		
28	Communication failure		
31	Other error		

4.2.2. CONTACT & CONTACT LESS READER API SPECIFICATION FOR SMART CARD ACCEPTER/DISPENSER

Contact R/W for contact SAM and contact less RFID R/W functionality related methods are included in this sub-section.

4.2.2.1. Connect Device

Purpose	To connect to the RFID Reader integrated with Smart Card Accepter/Dispenser if the Reader uses a different port other than the same port being used for Smart Card Accepter/Dispenser.		
Package Name	Cris	**Class Name**	SmartCard
Method Signature	int ConnectDevice(int PortId, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
PortId	Serial Port Number to which the device (RFID Reader) is connected	0-50	If the device uses a serial port.
		51	In case the device uses USB port
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values			
Value	**Description**		
0	Device connected successfully		
18	Operation timeout occurred		
20	Device already connected		
25	Port doesn't exist		
28	Communication failure		
31	Other error		

Caution

This method should be implemented only if Smart Card Reader is accessed through a different communication port than Smart Card Accepter/Dispenser device.

4.2.2.2. Get Smart Card Reader Native Lib Version

Purpose	To get the Native Library Version		
Package Name	Cris	**Class Name**	SmartCard
Method Signature	String GetSCardReaderNativeLibVersion()		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**

Return Values : Version information comprising the following: **MAJOR -** Incompatible API changes **MINOR** - Functionality adding in a backwards-compatible manner **BUILD** - Bug fixing backwards-compatible	
Value	**Description**
00.00.00	Error retreiving version information
01.00.00	A valid version string

Caution

This method should be implemented only if Smart Card Reader is accessed through a different communication port than Smart Card Accepter/Dispenser device.

4.2.2.3. Get Smart Card Reader FW Version

Purpose	To get the Reader FW Version		
Package Name	Cris	**Class Name**	SmartCard
Method Signature	String GetSCardReaderFWVersion()		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Return Values : Version information			
Value	**Description**		
00.00.00	Error retreiving version information		
01.00.00	A valid version string in the format or equivalent as received by the device		

Caution

This method should be implemented only if Smart Card Reader is accessed through a different communication port than Smart Card Accepter/Dispenser device.

4.2.2.4. Power On/Off Contact card Socket

Purpose	Purpose of the API is to power on or off a specific SAM Slot/socket.		
Package Name	Cris	**Class Name**	SmartCard
Method Signature	int SAMSlotPowerOnOff(intSAMSlotId,intPowerOnOffState, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
SAMSlotId	It indicates contact slot id which needs to be activated.	1	First SAM Slot
		2	Second SAM Slot

PowerOnOffState	Activation or deactivation of ISO-7816 Contact slots.	0	Power Off/Deactivate
		1	Power On/Activate
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values			
Value	**Description**		
0	Operation succeeded		
1	Operation failed		
18	Operation timeout occurred		
20	Device not yet connected		
28	Communication failure		
31	Other error		

4.2.2.5. Reset Contact card (SAM)

Purpose	Purpose of the API is to reset SAM in cold or worm mode.		
Package Name	Cris	**Class Name**	SmartCard
Method Signature	byte [] ResetSAM(intSAMSlotId, intResetType,int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
SAMSlotId	It indicates contact slot id where the SAM is inserted	1	First SAM Slot
		2	Second SAM Slot
ResetType	Werther it is a cold reset or worm reset.	0	Cold Reset
		1	Worm Reset
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values : Command execution status and ATR to be returned as a byte array as defined below:			
Value	**Description**		
Byte0	**Status of Reset**		
0	Operation successful		
1	Operation failed		
2	No contact card (SAM) found		
18	Operation timeout occurred		
20	Device not yet connected		
28	Communication failure		
31	Other error		
Byte1-n	Holds the ATR		

4.2.2.6. Activate Card

Purpose	Purpose of this API is to activate a contactless card (specifically MIFARE DESFire EV1 & MIFARE Ultralight card) and contact card (specifically MIFARE SAM AV1 & AV2) for read/write.		
Package Name	Cris	**Class Name**	SmartCard
Method Signature	byte [] ActivateCard(intCardTechType,intSAMSlotId, int Timeout)		
Input Parameters			

Parameter	**Description**	**Value/ Example**	**Meaning**
CardTechType	It indicates a contact less or contact card;	0	Contact less Card
		1	Contact Card
SAMSlotId	It indicates contact slot id of the target SAM which needs to be activated. In case of contact less card i.e.CardTechType=0 value of the SAMSlotId input parameter is always 0. Depending on the no. of Contact SAM slot/sockets the value of the SAMSlotId input parameter may vary.	1	First SAM Slot
		2	Second SAM Slot
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		

Return Values : Card activation status and type of contact or contactless card, size of UID and UID to be returned as a byte array as defined below:

Value	**Description**
Byte0	**Status of card activation**
0	Card found and activated
1	Card found but activation failed
2	Card found but it is unsupported
10	No card found
18	Operation timeout occurred
20	Device not yet connected
28	Communication failure
31	Other error
Byte1	**Type of card found**
	In case of Contactless Card
1	MIFARE DESFire
2	MIFARE DESFire EV1
3	MIFARE Ultralight
	In case of Contact Card
1	MIFARE SAM AV1
2	MIFARE SAM AV2
Byte2	Size of UID
Byte3-9	UID bytes

Note

Activation of MIFARE DESFire EV1 should include ISO 14443A-3 level commands REQA, AN1 & SELECT Cascade Level 1, AN2 & SELECT Cascade Level 2 followed by ISO 14443-4 level RATS & PPS commands.
Activation of MIFARE Ultralight card should include ISO 14443A-3 level commands REQA, AN1 & SELECT Cascade Level 1, AN2 & SELECT Cascade Level in order to make it ready for read/write operations.
Activation of contact card MIFARE SAM AV2 would mean necessary Reset of SAM followed by transmission of PPS command to select transmission factors F & D and protocol T=1 in order to make the SAM ready for ISO 7816-4 compliant APDU exchange.

4.2.2.7. Deactivate Card

Purpose	Purpose of this API is to deactivate an already activated contactless card (specifically MIFARE DESFire EV1 card) and contact card (specifically MIFARE SAM AV1 & AV2).		
Package Name	Cris	**Class Name**	SmartCard
Method Signature	int DeactivateCard(intCardTechType,intSAMSlotId,int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
CardTechType	It indicates a contact less or contact card;	0	Contact less Card
		1	Contact Card
SAMSlotId	It indicates contact slot id of the target SAM which needs to be activated. In case of contact less card i.e.CardTechType=0 value of the SAMSlotId input parameter is always 0. Depending on the no. of Contact SAM slot/sockets the value of the SAMSlotId input parameter may vary.	1	First SAM Slot
		2	Second SAM Slot
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values : Card deactivation status as defined below:			

Value	**Description**
0	Card found and deactivated
1	Card found but deactivation failed
10	No card found
18	Operation timeout occurred
20	Device not yet connected
28	Communication failure
31	Other error

Note

API should support exchanging APDUs with both contact and contactless cards simultaneously if both are activated. Activation of one type of card should not affect the activation status of the other one to allow interleaved APDU exchanges with activated contact and contact less cards.

4.2.2.8. Exchange APDU

Purpose	This API will allow to exchange ISO/IEC 14443-4 T=CL frame in native mode (i.e. DESFire Native APDU) and ISO/IEC 7816 wrapped APDU frame for contact less smart card (e.g. MIFARE DESFire EV1) and ISO7816 frame for contact card (MIFARE SAM AV1 &AV2).		
Package Name	Cris	**Class Name**	SmartCard
Method Signature	byte [] XChangeAPDU(intCardTechType, byte [] CommandAPDU, intSAMSlotId, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
CardTechType	It indicates a contact less or contact card;	0	Contact less Card
		1	Contact Card
CommandAPDU	ISO/IEC 14443-4 standard T=CL command APDU frame in native mode (i.e. DESFire Native APDU) or ISO/IEC 7816 wrapped APDU frame for contact less smart card and ISO 7816 command APDU frame for contact smart card wrapped in a byte array.		
SAMSlotId	It indicates contact slot id of the target SAM which needs to be activated. In case of contact less card i.e.CardTechType=0 value of the SAMSlotId input parameter is always 0. Depending on the no. of Contact SAM slot/sockets the value of the SAMSlotId input parameter may vary.	1	First SAM Slot
		2	Second SAM Slot
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values ResponseAPDU: Its a byte array containing execution status followed by ISO/IEC 14443-4 standard T=CL response APDU frame in native mode (i.e. DESFire Native APDU) or ISO/IEC 7816 wrapped APDU frame for contact less smart card and ISO 7816 response APDU frame for contact smart card.			
Value	**Description**		
Byte0	**Indicates status of command execution**		
0	Executed successfully		
1	Execution failed		
10	No card found		
18	Operation timeout occurred		
20	Device not yet connected		
28	Communication failure		
31	Other error		

Byte1-n	ISO/IEC 14443-4 standard T=CL response APDU frame in native mode (i.e. DESFire Native APDU) or ISO/IEC 7816 wrapped APDU frame for contact less smart card and ISO 7816 response APDU frame for contact smart card.

4.2.2.9. Read Ultralight Block

Purpose	Purpose of the API is to read 16 bytes data starting from a ultralight page address of MIFARE Ultralight.		
Package Name	Cris	**Class Name**	SmartCard
Method Signature	byte[] ReadUltralightBlock(int Addr, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Addr	Ultralight page address from where 16 bytes data to be read		
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values : The API will return Read status along with 16 bytes data in a byte array; where			
Value	**Description**		
Byte0	**Reading status**		
0	Reading successful		
1	Reading failed		
18	Operation timeout occurred		
20	Device not yet connected		
28	Communication failure		
31	Other error		
Byte1-16	16 bytes data (16 bytes should be filled with 0's in case of any error)		

4.2.2.10. Write Ultralight Page

Purpose	Purpose of the API is to write 4 bytes (one ultralight page) data to a specific ultralight page address of MIFARE Ultralight.		
Package Name	Cris	**Class Name**	SmartCard
Method Signature	int WriteUltralightPage(int Addr, byte [] Data, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**

Addr	Ultralight page address from where 4 bytes data to be written		
Data	4 bytes data to be written		
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		

Return Values : The API will return status of write operation; where

Value	Description
0	Write successful
1	Write failed
18	Operation timeout occurred
20	Device not yet connected
28	Communication failure
31	Other error

4.2.2.11. Disconnect Device

Purpose	This API will be used to disconnect the device [RFID Reader] if the Reader uses different port other than the same port being used for Smart Card Accepter/Dispenser.		
Package Name	Cris	**Class Name**	SmartCard
Method Signature	int DisConnectDevice(int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values			
Value	**Description**		
0	Disconnected successfully		
18	Operation timeout occurred		
20	Device not yet connected		
28	Communication failure		
31	Other error		

Caution

This method should be implemented only if Smart Card Reader is accessed through a different communication port than Smart Card Accepter/Dispenser device.

4.3. TOKEN DISPENSER & CONTACT AND CONTACTLESS R/W API SPECIFICATION

The methods implementing the functionalities of the Token Dispenser and integrated Contact and Contactless (RFID) R/Ws are listed in this section.

4.3.1.DISPENSER API

Token Dispenser functionality related methods are included in this sub-section.

4.3.1.1. Connect Device

Purpose	To connect to the Token Dispenser & RFID Reader combo device or Only the Token Dispenser device.		
Package Name	Cris	**Class Name**	TokenDispenser
Method Signature	int ConnectDevice(int PortId, int ChannelClearanceMode, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
PortId	Serial Port Number to which the device is connected	0-50	If the device uses a serial port.
		51	In case the device uses USB port
ChannelClearanceMode	If there is a Token in the device channel or at staging area, either it will be sent to rejection bin or the dispensing outlet or kept in its position.	0	Retain in the channel
		1	Send to rejection bin
		2	Send to dispensing outlet of the device
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		

Return Values	
Value	**Description**
0	Device connected successfully
1	Channel clearance failed due to rejection bin is full
2	Channel clearance failed due to channel is blocked
3	Channel clearance failed due to unknown reason
18	Operation timeout occurred
20	Device already connected (Main Module)
25	Port doesn't exist (Main Module)
26	Port doesn't exist (Reader)
28	Communication failure (Main Module)
29	Communication failure (Reader)
31	Other error

Note

In case the combo device uses separate ports for Token Dispenser and RFID Reader(s), RFID Reader(s) and Contact Card (SAM) to be connected using the method described under section **4.3.2.1 ConnectDevice.**

4.3.1.2. Get Native Lib Version

Purpose	To get the Native Library Version		
Package Name	Cris	**Class Name**	TokenDispenser
Method Signature	String GetNativeLibVersion()		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Return Values : Version information comprising the following: **MAJOR -** Incompatible API changes **MINOR -** Functionality adding in a backwards-compatible manner **BUILD -** Bug fixing backwards-compatible			
Value	**Description**		
00.00.00	Error retreiving version information		
01.00.00	A valid version string		

4.3.1.3. Get Token Dispenser FW Version

Purpose	To get the Token Dispenser FW Version		
Package Name	Cris	**Class Name**	TokenDispenser
Method Signature	String GetTokenDispenserFWVersion()		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Return Values : Version information			
Value	**Description**		
00.00.00	Error retreiving version information		
01.00.00	A valid version string in the format or equivalent as received by the device		

4.3.1.4. Get Token Dispenser Reader FW Version

Purpose	To get the Token Dispenser Reader FW Version		
Package Name	Cris	**Class Name**	TokenDispenser
Method Signature	String GetTokenDispenserReaderFWVersion()		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Return Values : Version information			
Value	**Description**		

00.00.00	Error retreiving version information
01.00.00	A valid version string in the format or equivalent as received by the device

Caution

This method should be implemented only if Smart Card Reader shares the same communication port as Token dispenser device.

4.3.1.5. Device Status

Purpose	To collect status of different components of the device		
Package Name	Cris	**Class Name**	TokenDispenser
Method Signature	byte[] GetDeviceStatus(int ComponentId, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
ComponentId *	Identifier of the device sub-components	0	All components
		1	Reader
		2	Token Cointainer
		3	Rejection Bin
		4	Channel
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values : Execution status of the API and status of the device components to be returned as a byte array as defined below:			
Value	**Description**		
Byte0	**Execution status of the API**		
0	Operation successful		
1	Operation failed		
18	Operation timeout occurred		
20	Device not yet connected		
28	Communication failure (Main Module)		
29	Communication failure (Reader)		
31	Other error		
Byte1	**RFID Reader status**		
0	Ready		
1	Not Ready		
Byte2	**SAM Reader status**		
0	Ready		
1	Not Ready		
Byte3	**Token Container Status**		
0	Empty		

1	Not empty
Byte4	**Rejection Bin status**
0	Full
1	Not full
Byte5	**Channel status**
0	Clear
1	Blocked
Byte6-7	**Channel sensor status **** 16 bits where each bit will indicate whether a sensor is blocked or not; If no of sensors is less than 16, higher significant bits will be filled with 0's.
0	Clear
1	Blocked

Note

* When ComponentId is other than 0, API will fill relevant values for the requested component leaving other component status 0; however execution status of the API is mandatory for any value of ComponentId.

Note

** Supplier of the APIs must share the relevant sensor information including the interpretation of each bit of the "Channel sensor status" bytes along with the submission of the APIs.

Note

This API will be invoked frequently even when the devices are busy in dispensing Tokens during the life cycle of calling application.

4.3.1.6. Dispense Token (Phase1)

Purpose	It will dispense a token from Token Container and place it at the staging area for RFID read/Write operation. The API should not dispense any token if there is a Token at staging area or anywhere in the channel. If within the predefined time period the API doesn't find any token or any blocking in the channel is detected, it will return with relevant error code as specified in return value of the API.		
Package Name	Cris	**Class Name**	TokenDispenser
Method Signature	int DispenseTokenPhase1(int BoxNo, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
BoxNo	Identifier of the Token Container from which a token to be dispensed.	1	Box1
		2	Box2
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values			
Value	**Description**		
0	Operation successful		

1	Operation failed
2	Channel blocked
3	Specified Box is empty
18	Operation timeout occurred
20	Device not yet connected
28	Communication failure (Main Module)
31	Other error

If there is only one Box/Container with the Token Dispenser device, BoxNo parameter will be set to 1 i.e. it is always Box1.

4.3.1.7. Dispense Token (Phase2)

Purpose	It will dispense a token from Staging area to dispensing outlet or rejection bin after RFID read/Write operation. If within the predefined time period the API doesn't find any token at the staging area, it will return with relevant error code as specified in return value of the API.		
Package Name	Cris	**Class Name**	TokenDispenser
Method Signature	int DispenseTokenPhase2(int BoxNo, int TokenDest, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
BoxNo	Identifier of the Token Container of the corresponding staging area from which a token to be dispensed.	1	Box1
		2	Box2
TokenDest	Destination of the Token to which the Token to be dispensed.	1	Dispensing outlet
		2	Rejection Bin
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		

Return Values	
Value	**Description**
0	Operation successful
1	Operation failed
2	Channel blocked
3	No Token found at the Staging area
18	Operation timeout occurred
20	Device not yet connected
28	Communication failure (Main Module)
31	Other error

Note

If there is only one Box/Container with the Token Dispenser device or Two Box/Container is connected to a single staging area, BoxNo parameter will be set to 1

4.3.1.8. Empty Token Box

Purpose	It will dispense all the tokens from a Token container to dispensing outlet or rejection. If within the predefined time period the API doesn't find any token in the Token Container, it will return with relevant error code as specified in return value of the API.		
Package Name	Cris	**Class Name**	TokenDispenser
Method Signature	int EmptyTokenBox(int BoxNo, int TokenDest, int Timeout)		
Input Parameters			

Parameter	**Description**	**Value/ Example**	**Meaning**
BoxNo	Identifier of the Token Container to be emptied.	1	Box1
		2	Box2
TokenDest	Destination of the Token to which the Token to be dispensed.	1	Dispensing outlet
		2	Rejection Bin
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		

Return Values	
Value	**Description**
0	Operation successful
1	Operation failed
2	Channel blocked
3	Token container is already empty
18	Operation timeout occurred
20	Device not yet connected
28	Communication failure (Main Module)
31	Other error

Note

If there is only one Box/Container with the Token Dispenser device, BoxNo parameter will be set to 1 i.e. it is always Box1.

4.3.1.9. Clear Jammed Token

Purpose	It will clear any jammed token and send it to dispensing outlet or rejection bin. If within the predefined time period the API doesn't find any jammed token anywhere in the channel, it will return with relevant error code as specified in return value of the API.		
Package Name	Cris	**Class Name**	TokenDispenser
Method Signature	int ClearJammedToken(int BoxNo, int TokenDest, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
BoxNo	Identifier of the Token Container of the corresponding channel to be cleared.	1	Box1
		2	Box2
TokenDest	Destination of the Tokens to which the jammed Token to be sent	1	Dispensing outlet
		2	Rejection Bin
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values			
Value	**Description**		
0	Operation successful		
1	Operation failed		
2	Channel blocked		
3	No token found in Channel		
18	Operation timeout occurred		
20	Device not yet connected		
28	Communication failure (Main Module)		
31	Other error		

Note

If there is only one Box/Container with the Token Dispenser device, BoxNo parameter will be set to 1 i.e. it is always Box1.

4.3.1.10. Disconnect Device

Purpose	This API will be used to disconnect the device.		
Package Name	Cris	**Class Name**	TokenDispenser
Method Signature	int DisConnectDevice(int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Timeout	Time in milliseconds the API will try to perform its intended operation		

	otherwise return timeout status.		
Return Values			
Value	**Description**		
0	Disconnected successfully		
1	Disconnected successfully but a token is in the channel		
18	Operation timeout occurred		
20	Device not yet connected		
28	Communication failure (Main Module)		
31	Other error		

4.3.2.CONTACT & CONTACT LESS READER API FOR TOKEN DISPENSER

Contact R/W for contact SAM and contact less RFID R/W functionality related methods are included in this sub-section.

4.3.2.1. Connect Device

Purpose	To connect to the RFID Reader(s) integrated with Token Dispenser if the Reader(s) uses different port other than the same port being used for TokenDispenser.		
Package Name	Cris	**Class Name**	TokenDispenser
Method Signature	int ConnectDevice(int DeviceId, int PortId, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
DeviceId	Device Id of the RFID Readers integrated with the Token Dispenser Unit.	1	First Reader device
		2	Second Reader device
PortId	Serial Port Number to which the device (RFID Reader) is connected	0-50	If the device uses a serial port.
		51	In case the device uses USB port
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values			
Value	**Description**		
0	Device connected successfully		
18	Operation timeout occurred		
20	Device already connected		
25	Port doesn't exist		
28	Communication failure		
31	Other error		

Note

If there is only one RFID Reader with the Token Dispenser device, DeviceId parameter will be set to 1 i.e. it is always "First Reader device".

Caution

This method should be implemented only if Smart Card Reader is accessed through a different communication port than Token Dispenser device.

4.3.2.2. Get Token Dispenser Reader FW Version

Purpose	To get the Token Dispenser FW Version		
Package Name	Cris	**Class Name**	TokenDispenser
Method Signature	String GetTokenDispenserReaderFWVersion()		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Return Values : Version information			
Value	**Description**		
00.00.00	Error retreiving version information		
01.00.00	A valid version string in the format or equivalent as received by the device		

Caution

This method should be implemented only if Smart Card Reader is accessed through a different communication port than Token Dispenser device.

4.3.2.3. Power On/Off Contact card Socket

Purpose	Purpose of the API is to power on or off a specific SAM Slot/socket.		
Package Name	Cris	**Class Name**	TokenDispenser
Method Signature	int SAMSlotPowerOnOff(int DeviceId, intSAMSlotId,intPowerOnOffState, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
DeviceId	Device Id of the RFID Readers integrated with the Token Dispenser Unit.	1	First Reader device
		2	Second Reader device
SAMSlotId	It indicates contact slot id which needs to be reset..	1	First SAM Slot
		2	Second SAM Slot
PowerOnOffState	Activation or deactivation of ISO-7816 Contact slots.	0	Power Off/Deactivate
		1	Power On/Activate

Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values			
Value	**Description**		
0	Operation succeeded		
1	Operation failed		
18	Operation timeout occurred		
20	Device not yet connected		
28	Communication failure		
31	Other error		

If there is only one RFID Reader with the Token Dispenser device, DeviceId parameter will be set to 1 i.e. it is always "First Reader device".

4.3.2.4. Reset Contact Card (SAM)

Purpose	Purpose of the API is to reset SAM in cold or worm mode.		
Package Name	Cris	**Class Name**	TokenDispenser
Method Signature	byte [] ResetSAM(int DeviceId, SAMSlotId, intResetType,int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
DeviceId	Device Id of the RFID Readers integrated with the Token Dispenser Unit.	1	First Reader device
		2	Second Reader device
SAMSlotId	It indicates contact slot id which needs to be reset.	1	First SAM Slot
		2	Second SAM Slot
ResetType	Werther it is a cold reset or worm reset	0	Cold Reset
		1	Worm Reset
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values : Command execution status and ATR to be returned as a byte array as defined below:			
Value	**Description**		
Byte0	**Status of Reset**		
0	Operation successful		
1	Operation failed		
2	No contact card (SAM) found		
18	Operation timeout occurred		
20	Device not yet connected		
28	Communication failure		

31	Other error
Byte1-n	Holds the ATR

If there is only one RFID Reader with the Token Dispenser device, DeviceId parameter will be set to 1 i.e. it is always "First Reader device".

4.3.2.5. Activate Card

Purpose	Purpose of this API is to activate a contactless token (specifically MIFARE Ultralight) and contact card (specifically MIFARE SAM AV1 & AV2) for read/write.		
Package Name	Cris	**Class Name**	TokenDispenser
Method Signature	byte [] ActivateCard(int DeviceId, int CardTechType,int SAMSlotId, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
DeviceId	Device Id of the RFID Readers integrated with the Token Dispenser Unit.	1	First Reader device
		2	Second Reader device
CardTechType	It indicates a contact less Token or contact card;	0	Contact less Token
		1	Contact Card
SAMSlotId	It indicates contact slot id of the target SAM which needs to be activated. In case of contact less card i.e.CardTechType=0 value of the SAMSlotId input parameter is always 0. Depending on the no. of Contact SAM slot/sockets the value of the SAMSlotId input parameter may vary.	1	First SAM Slot
		2	Second SAM Slot
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values : Token activation status and type of contact or contactless token, size of UID and UID to be returned as a byte array as defined below:			
Value	**Description**		
Byte0	Status of token activation		
0	Card found and activated		
1	Card found but activation failed		
2	Card found but it is unsupported		
10	No card found		
18	Operation timeout occurred		
20	Device not yet connected		
28	Communication failure		
31	Other error		
Byte1	**Type of token found**		

	In case of Contactless Token
1	MIFARE DESFire
2	MIFARE DESFire EV1
3	MIFARE Ultralight
	In case of Contact Card
1	MIFARE SAM AV1
2	MIFARE SAM AV2
Byte2	Size of UID
Byte3-9	UID bytes

Note

Activation of MIFARE Ultralight Token should include ISO 14443A-3 level commands REQA, AN1 & SELECT Cascade Level 1, AN2 & SELECT Cascade Level in order to make it ready for read/write operations.
Activation of contact card MIFARE SAM AV2 would mean necessary Reset of SAM followed by transmission of PPS command to select transmission factors F & D and protocol T=1 in order to make the SAM ready for ISO 7816-4 compliant APDU exchange.

If there is only one RFID Reader with the Token Dispenser device, DeviceId parameter will be set to 1 i.e. it is always "First Reader device".

4.3.2.6. Deactivate Card

Purpose	Purpose of this API is to deactivate an already activated contactless token (specifically MIFARE Ultralight Token) and contact card (specifically MIFARE SAM AV1 & AV2).		
Package Name	Cris	**Class Name**	TokenDispenser
Method Signature	int DeactivateCard(int DeviceId, intCardTechType,intSAMSlotId,int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
DeviceId	Device Id of the RFID Readers integrated with the Token Dispenser Unit.	1	First Reader device
		2	Second Reader device
CardTechType	It indicates a contact less token or contact card.	0	Contact less Token
		1	Contact Card (SAM)
SAMSlotId	It indicates contact slot id of the target SAM which needs to be activated. In case of contact less card i.e.CardTechType=0 value of the SAMSlotId input parameter is always 0. Depending on the no. of Contact SAM slot/sockets the value of the SAMSlotId input parameter may vary.	1	First SAM Slot
		2	Second SAM Slot
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		

Return Values : Token deactivation status as defined below:	
Value	**Description**
0	Card found and deactivated
1	Card found but deactivation failed
10	No card found
18	Operation timeout occurred
20	Device not yet connected
28	Communication failure
31	Other error

Note

If there is only one RFID Reader with the Token Dispenser device, DeviceId parameter will be set to 1 i.e. it is always "First Reader device".

4.3.2.7. Read Ultralight Block

Purpose	Purpose of the API is to read 16 bytes data starting from a ultralight page address of MIFARE Ultralight.		
Package Name	Cris	**Class Name**	TokenDispenser
Method Signature	byte[] ReadUltralightBlock(int DeviceId, int Addr, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
DeviceId	Device Id of the RFID Readers integrated with the Token Dispenser Unit.	1	First Reader device
		2	Second Reader device
Addr	Ultralight page address from where 16 bytes data to be read		
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values : The API will return Read status along with 16 bytes data in a byte array; where			
Value	**Description**		
Byte0	Read status		
0	Reading successful		
1	Reading failed		

18	Operation timeout occurred
20	Device not yet connected
28	Communication failure
31	Other error
Byte1-16	16 bytes data (16 bytes should be filled with 0's in case of any error)

If there is only one RFID Reader with the Token Dispenser device, DeviceId parameter will be set to 1 i.e. it is always "First Reader device".

4.3.2.8. Write Ultralight Page

Purpose	Purpose of the API is to write 4 bytes (one ultralight page) data to a specific ultralight page address of MIFARE Ultralight.		
Package Name	Cris	**Class Name**	TokenDispenser
Method Signature	int WriteUltralightPage(int DeviceId, int Addr, byte [] Data, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
DeviceId	Device Id of the RFID Readers integrated with the Token Dispenser Unit.	1	First Reader device
		2	Second Reader device
Addr	Ultralight page address from where 4 bytes data to be written.		
Data	4 bytes data to be written		
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values : The API will return status of write operation; where			
Value	**Description**		
0	Write successful		
1	Write failed		
18	Operation timeout occurred		
20	Device not yet connected (Main Module)		
28	Communication failure		
31	Other error		

Note

If there is only one RFID Reader with the Token Dispenser device, DeviceId parameter will be set to 1 i.e. it is always "First Reader device".

4.3.2.9. Disconnect Device

Purpose	This API will be used to disconnect the device(s) [RFID Reader(s)] if the Reader(s) uses different port other than the same port being used for TokenDispenser.		
Package Name	Cris	**Class Name**	TokenDispenser
Method Signature	int DisConnectDevice(int DeviceId, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
DeviceId	Device Id of the RFID Readers integrated with the Token Dispenser Unit.	1	First Reader device
		2	Second Reader device
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values			
Value	**Description**		
0	Disconnected successfully		
1	Disconnected successfully but a token is in the channel		
18	Operation timeout occurred		
20	Device not yet connected		
28	Communication failure (Main Module)		
31	Other error		

Note

If there is only one RFID Reader with the Token Dispenser device, DeviceId parameter will be set to 1 i.e. it is always "First Reader device".

Caution

This method should be implemented only if Smart Card Reader is accessed through a different communication port than Token Dispenser device.

4.4. CURRENCY NOTE/COIN ACCEPTER DEVICE API SPECIFICATION

The methods implementing the functionalities of the Currency Note and Coin Accepters are listed in this section.

4.4.1. Connect Device

Purpose	To connect the Currency Note/Coin Accepter devices.		
Package Name	Cris	**Class Name**	Currency
Method Signature	int ConnectDevice(int PortId1, int PortId2, int PortId3, int DeviceType, int EscrowClearanceMode, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
PortId1	Serial Port Number to which the Note Accepter device is connected.	0-50	If note accepter uses a serial port.
		51	In case note accepter uses USB port
PortId2	Serial Port Number to which the Coin Accepter device is connected	0-50	A valid serial port id
PortId3	Serial Port Number to which the Coin Escrow device is connected	0-50	If coin escrow uses a separate serial port.
		51	In case coin escrow doesn't use separate port
DeviceType	Type of device(s) to be connected	1	Only Currency Note Accepter
		2	Only Currency Coin Accepter with escrow
EscrowClearanceMode	If there is any currency note/coin left in the escrow, either it will be sent to collection bin or kept in its position.	0	Retain escrowed note(s)/coin(s) in the escrow
		1	Send the escrowed note(s)/ coin(s) in the collection bin.
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values			
Value	**Description**		
0	Device connected successfully		
1	Device connected successfully with few notes/coins left in escrow		
2	Security door is opened		
3	Escrow clearance failed due to collection bin full		
4	Escrow clearance failed due to any blocking in Note/Coin Accepter channel		
5	Escrow clearance failed due to unknown reason		
18	Operation timeout occurred		
20	Note Accepter already connected		
21	Coin Accepter already connected		
22	Coin Escrow already connected		
25	Note Accepter Port doesn't exist		
26	Coin Accepter or escrow Port doesn't exist		

27	Note and Coin Accepter or escrow Ports doesn't exist
28	Note Accepter Communication failure
29	Coin Accepter Communication failure
30	Note and Coin Accepter Communication failure
31	Other error

4.4.2.Get Native Lib Version

Purpose	To get the Native Library Version		
Package Name	Cris	**Class Name**	Currency
Method Signature	String GetNativeLibVersion()		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Return Values : Version information comprising the following: **MAJOR -** Incompatible API changes **MINOR -** Functionality adding in a backwards-compatible manner **BUILD -** Bug fixing backwards-compatible			
Value	**Description**		
00.00.00	Error retreiving version information		
01.00.00	A valid version string		

4.4.3.Get Currency Device FW Version

Purpose	To get the Currency Device FW Version		
Package Name	Cris	**Class Name**	Currency
Method Signature	String GetCurrencyDevFWVersion(int CurrencyType)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
CurrencyType	Type of Currency	**1**	**Currency Note**
		2	**Currency Coin**
Return Values : Version information of Currency Note/Coin			
Value	**Description**		
00.00.00	Error retreiving version information		
01.00.00	A valid version string in the format or equivalent as received by the device		

4.4.4.Device Status

Purpose	Purpose of this API is to collect the status or state of the currency device(s) and their sub-components along with escrowed note/coin counts.		
Package Name	Cris	**Class Name**	Currency
Method Signature	byte[] DeviceStatus(int DeviceType, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
DeviceType	Identifier of the device sub-components	0	Both the Currency Note Accepter and Currency Coin Accepter with Coin escrow.
		1	Only the Currency Note Accepter
		2	Only Currency Coin Accepter with Coin escrow
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values : LSB of the byte array will signify state, status of different components of the device and communication status. Some of the mandatory return state/status are as given below :			
Value	**Description**		
Byte0	**Execution status of the API**		
0	Operation successful		
1	Operation failed		
18	Operation timeout occurred		
20	Note Accepter not yet connected		
21	Coin Accepter not yet connected		
22	Coin Escrow not yet connected		
28	Communication failure (Note Accepter)		
29	Communication failure (Coin Accepter)		
30	Communication failure (Coin Escrow)		
31	Other error		
Byte1	**Note Accepter status**		
Byte1:0	**Note Accepter Communication status**		
0	Ready		
1	Not Ready		
Byte1:1	**Note Accepter ~~Readyness~~ Readiness**		
0	Ready		
1	Not Ready		
Byte1:2	**Security Door status**		
0	Opened		
1	Closed		
Byte1:3	**Escrow status**		
0	Empty		

1	Not empty
Byte1:4	**Collection Box status***
0	Full
1	Not full
Byte1:5	**Insertion Slot status**
0	Clear
1	Blocked
Byte1:6	**Transport Channel status**
0	Clear
1	Blocked
Byte1:7	RFU
Byte2	**Coin Accepter status**
Byte2:0	**Coin Accepter and Escrow Communication status**
0	Ready
1	Not Ready
Byte2:1	**Coin Accepter Readyness**
0	Ready
1	Not Ready
Byte2:2	**Security Door status**
0	Opened
1	Closed
Byte2:3	**Escrow status**
0	Empty
1	Not empty
Byte2:4	**Collection Box status***
0	Full
1	Not full
Byte2:5	**Insertion Slot status**
0	Clear
1	Blocked
Byte2:6	**Transport Channel status**
0	Clear
1	Blocked
Byte2:7	RFU
Byte3-7	**Escrowed Notes**
Byte3:0-3	Indicates no of escrowed INR 5 Notes
Byte3:4-7	Indicates no of escrowed INR 10 Notes
Byte4:0-3	Indicates no of escrowed INR 20 Notes
Byte4:4-7	Indicates no of escrowed INR 50 Notes
Byte5:0-3	Indicates no of escrowed INR 100 Notes
Byte5:4-7	Indicates no of escrowed INR 200 Notes
Byte6:0-3	Indicates no of escrowed INR 500 Notes

Byte6:4-7	Indicates no of escrowed INR 1000 Notes
Byte7:0-3	Indicates no of escrowed INR 2000 Notes
Byte7:4-7	RFU
Byte8	RFU
Byte9-11	**Escrowed Coins**
Byte9:0-3	RFU
Byte9:4-7	RFU
Byte10:0-3	Indicates no of escrowed INR 5 Coins
Byte10:4-7	Indicates no of escrowed INR 10 Coins
Byte11:0-3	RFU
Byte11:4-7	RFU


Note

This API will be invoked frequently even when the devices are busy in accepting currency notes/coins during the life cycle of calling application.

4.4.5.Get Valid Currency

Purpose	This API will synchronously enable the accepter to validate a single Currency Note/Coin and wait for a Currency Note/Coin if the device is ready to accept a Currency Note/Coin. After validating a Currency Note/Coin or waiting for a Currency Note/Coin till the timeout, it will wait for a AcceptCurrentCurrency API call and keep its acceptance state until StackAcceptedCurrency or ReturnAcceptedCurrency API is called. Application may call this API several times as per requirement in a single cash transaction.		
Package Name	Cris	**Class Name**	Currency
Method Signature	int GetValidCurrency(int CurrencyType, int Denom, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
CurrencyType	Type of Currency	1	Currency Note
		2	Currency Coin
Denom	Denomination of the Currency Note/Coin	5	INR 5
		10	INR 10
		..	.....
		2000	INR 2000
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values : An int indicating one of the following:			
Value	**Description**		
0	Note/Coin of correct denomination validated		

1	Note/Coin rejected due to insertion of an invalid (mutilated/soiled/non-gandhi series note/counterfeit) note or a coin
2	Note/Coin rejected due to insertion of a inhibited denomination
3	Required Currency Accepter is not ready
18	Operation timeout occurred
20	Note Accepter not yet connected
21	Coin Accepter not yet connected
22	Coin Escrow not yet connected
28	Communication failure (Note Accepter)
29	Communication failure (Coin Accepter)
30	Communication failure (Coin Escrow)
31	Other error


Note

The API should ensure the healthy status of all the involved devices and their sub-components before executing its intended operation.

4.4.6.Accept Current Currency

Purpose	This API will accept an already validated Currency Note/Coin in escrow.		
Package Name	Cris	**Class Name**	Currency
Method Signature	int AcceptCurrentCurrency(int CurrencyType , int Denom, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
CurrencyType	Type of Currency	1	Currency Note
		2	Currency Coin
Denom	Denomination of the Currency Note/Coin		
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values : An int indicating one of the following:			
Value	**Description**		
0	Note/Coin of correct denomination accepted		
1	No currency Note/Coin is not yet validated		
2	Note escrow full		
3	Coin escrow full		
18	Operation timeout occurred		
20	Note Accepter not yet connected		
21	Coin Accepter not yet connected		
22	Coin Escrow not yet connected		
28	Communication failure (Note Accepter)		
29	Communication failure (Coin Accepter)		

30	Communication failure (Coin Escrow)
31	Other error


Note

The API should ensure the healthy status of all the involved devices and their sub-components before executing its intended operation.

4.4.7.Return Current Currency

Purpose	This API will return the currently validated Currency Note/Coin from escrow.		
Package Name	Cris	**Class Name**	Currency
Method Signature	int ReturnCurrentCurrency(int CurrencyType , int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
CurrencyType	Type of Currency	1	Currency Note
		2	Currency Coin
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values : An int indicating one of the following:			
Value	**Description**		
0	Note/Coin Returned		
1	Return mouth blocked (Note Accepter)		
2	Return mouth blocked (Note Accepter)		
18	Operation timeout occurred		
20	Note Accepter not yet connected		
21	Coin Accepter not yet connected		
22	Coin Escrow not yet connected		
28	Communication failure (Note Accepter)		
29	Communication failure (Coin Accepter)		
30	Communication failure (Coin Escrow)		
31	Other error		


Note

The API should ensure the healthy status of all the involved devices and their sub-components before executing its intended operation.

4.4.8.Enable Denominations

<table>
<tr><td>Purpose</td><td colspan="4">Purpose of this API is to activate denominations of Currency Notes/Coins to be accepted by the Currency validaters.</td></tr>
<tr><td>Package Name</td><td>Cris</td><td>Class Name</td><td colspan="2">Currency</td></tr>
<tr><td>Method Signature</td><td colspan="4">int EnableTheseDenominations(int CurrencyType, int DenomMask, int Timeout)</td></tr>
<tr><td colspan="5">Input Parameters</td></tr>
<tr><td>Parameter</td><td>Description</td><td colspan="2">Value/ Example</td><td>Meaning</td></tr>
<tr><td rowspan="3">CurrencyType</td><td rowspan="3">Type of Currency for which specific denomination(s) to be enabled or disabled for acceptance.</td><td colspan="2">0</td><td>Currency Note and Coin</td></tr>
<tr><td colspan="2">1</td><td>Currency Note</td></tr>
<tr><td colspan="2">2</td><td>Currency Coin</td></tr>
<tr><td rowspan="24">DenomMask</td><td rowspan="24">Each bit of Lower Significant Bytes of this java int type parameter will indicate whether a specific denomination is allowed or not.</td><td colspan="2">Representation of Mask Bit</td><td></td></tr>
<tr><td colspan="2">0</td><td>Not allowed/Inhibited</td></tr>
<tr><td colspan="2">1</td><td>Allowed/Enabled</td></tr>
<tr><td colspan="2">Setting of Mask Bits</td><td></td></tr>
<tr><td>Byte0-1</td><td>Mask Bytes for Note</td></tr>
<tr><td>Byte0:0</td><td>Indicates Note INR 5 is to be enabled/disabled</td></tr>
<tr><td>Byte0:1</td><td>Indicates Note INR 10 is to be enabled/disabled</td></tr>
<tr><td>Byte0:2</td><td>Indicates Note INR 20 is to be enabled/disabled</td></tr>
<tr><td>Byte0:3</td><td>Indicates Note INR 50 is to be enabled/disabled</td></tr>
<tr><td>Byte0:4</td><td>Indicates Note INR 100 is to be enabled/disabled</td></tr>
<tr><td>Byte0:5</td><td>Indicates Note INR 200 is to be enabled/disabled</td></tr>
<tr><td>Byte0:6</td><td>Indicates Note INR 500 is to be enabled/disabled</td></tr>
<tr><td>Byte0:7</td><td>Indicates Note INR 1000 is to be enabled/disabled</td></tr>
<tr><td>Byte1:0</td><td>Indicates Note INR 2000 is to be enabled/disabled</td></tr>
<tr><td>Byte1:1-7</td><td>RFU and bits are set to 0</td></tr>
<tr><td>Byte2</td><td>Mask Byte for Coin</td></tr>
<tr><td>Byte2:0</td><td>Indicates Coin INR 1 is to be enabled/disabled</td></tr>
<tr><td>Byte2:1</td><td>Indicates Coin INR 2 is to be enabled/disabled</td></tr>
<tr><td>Byte2:2</td><td>Indicates Coin INR 5 is to be enabled/disabled</td></tr>
</table>

		Byte2:3	Indicates Coin INR 10 is to be enabled/disabled
		Byte2:4-7	**RFU and bits are set to 0**
		Byte3	**RFU and bits are set to 0**
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		

Return Values : An int indicating one of the following:

Value	Description
0	Operation successful
1	Operation failed
18	Operation timeout occurred
20	Note Accepter not yet connected
21	Coin Accepter not yet connected
22	Coin Escrow not yet connected
28	Communication failure (Note Accepter)
29	Communication failure (Coin Accepter)
30	Communication failure (Coin Escrow)
31	Other error

4.4.9. Accept Currencies

Purpose	This API will asynchronously enable the accepters to accept Currency Notes/Coins if the devices are ready to accept Currency Note/Coin. It will return true immediately after bringing the device in acceptance state. Calling application will repeatedly call GetAcceptedAmount API to determine the total accepted amount so far and the current status of the asynchronous API.		
Package Name	Cris	**Class Name**	Currency
Method Signature	boolean AcceptCurrencies(int CurrencyType, int Amount, int Timeout)		

Input Parameters

Parameter	Description	Value/ Example	Meaning
CurrencyType	Type of Currency Note, Coin or both	0	Currency Note and Coin both are allowed
		1	Only Currency Note is allowed
		2	Only Currency Coin is allowed

Amount	Total amount to be accepted in Currency Notes, Currency Coins or Currency Notes and Coins depending upon the CurrencyType.		
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values : A boolean indicating one of the following:			
Value	**Description**		
True	Accepter(s) is(are) in acceptance state.		
False	One of the required Currency Accepters or both is(are) not ready to accept Currency.		

Note

If CurrencyType parameter is 1 or 2 then the respective Currency accepter needs to be ready for acceptance and in case of the parameter value is 0 both the Currency accepters need to be in ready state. Coin escrow should also be ready if the cash transaction involves a Coin Accepter.

The API should ensure the healthy status of all the involved devices and their sub-components before executing its intended operation.

4.4.10. Get Accepted Amount

Purpose	This API is intended to call periodically after calling asynchronous API AcceptCurrencies to determine the status of acceptance or total amount accepted in escrow within timeout period.		
Package Name	Cris	**Class Name**	Currency
Method Signature	int GetAcceptedAmount(byte[][] AcptdAmt)		
Input/Output Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
AcptdAmt	A 2D byte array (Size will be capacity of Escrow of the Bank Note Validator or Coin Validator whichever is larger) where each byte of the 0th 1D array indicates denomination of the accepted currency coins and each byte of the 1st 1D array indicates denomination of the accepted currency notes. API will fill the bytes in the 2D byte array after accepting each note/coin in escrow. otherwise return timeout status.		
Return Values : An integer indicating status of currency acceptance like-			
Value	**Description**		
0	Accepting		
1	Exact amount accepted		

2	Excess amount accepted
3	Note Escrow is full
4	Coin Escrow is full
5	Note and Coin Escrows are full
18	Operation timeout occurred
20	Note Accepter not yet connected
21	Coin Accepter not yet connected
22	Coin Escrow not yet connected
28	Communication failure (Note Accepter)
29	Communication failure (Coin Accepter)
30	Communication failure (Coin Escrow)
31	Other error

4.4.11. Stack Accepted Currencies

Purpose	Purpose of this API is to take the escrowed currency notes/coins in respective chamber of the note and coin stacker in the cash vault.		
Package Name	Cris	**Class Name**	Currency
Method Signature	int StackAcceptedCurrencies(int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values : An int indicating one of the following:			
Value	**Description**		
0	Note(s)/Coin(s) Stacked		
1	Note Stacker/Cash box is full		
2	Coin Stacker/Cash box is full		
3	Note and Coin Stacker/Cash box is full		
4	Note Transport Channel blocked		
5	Coin Transport Channel blocked		
6	Note and Coin Transport Channels are blocked		
18	Operation timeout occurred		
20	Note Accepter not yet connected		
21	Coin Accepter not yet connected		
22	Coin Escrow not yet connected		
28	Communication failure (Note Accepter)		
29	Communication failure (Coin Accepter)		
30	Communication failure (Coin Escrow)		
31	Other error		


Note

The API should ensure the healthy status of all the involved devices and their sub-components before executing its intended operation.

4.4.12. Return Accepted Currencies

Purpose	Purpose of this API is to return the escrowed currency notes/coins to the customer		
Package Name	Cris	**Class Name**	Currency
Method Signature	int ReturnAcceptedCurrencies(int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values : An int indicating one of the following:			

Value	**Description**
0	Note(s)/Coin(s) Returned
1	Note Return mouth blocked
2	Coin Return mouth blocked
3	Note and Coin Return mouth both are blocked
4	Note Transport Channel blocked
5	Coin Transport Channel blocked
6	Note and Coin Transport Channels are blocked
18	Operation timeout occurred
20	Note Accepter not yet connected
21	Coin Accepter not yet connected
22	Coin Escrow not yet connected
28	Communication failure (Note Accepter)
29	Communication failure (Coin Accepter)
30	Communication failure (Coin Escrow)
31	Other error


Note

The API should ensure the healthy status of all the involved devices and their sub-components before executing its intended operation.

4.4.13. Is Note Removed

Purpose	Purpose of the API is to determine whether customer has taken out the returned Notes. Calling application may call the API repeatedly to alert/instruct the customer to remove the returned notes from dispensing outlet.		
Package Name	Cris	**Class Name**	Currency
Method Signature	int IsNoteRemoved(int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values : An int indicating one of the following:			
Value	**Description**		
0	Notes Removed		
1	Notes not yet Removed		
18	Operation timeout occurred		
20	Note Accepter not yet connected		
21	Coin Accepter not yet connected		
22	Coin Escrow not yet connected		
28	Communication failure (Note Accepter)		
29	Communication failure (Coin Accepter)		
30	Communication failure (Coin Escrow)		
31	Other error		

4.4.14. Clear Jammed Currencies

Purpose	This API will help clearance of notes/coins jammed in escrow or anywhere in the transportation path. Depending on the Clearance mode, API will either stack jammed notes/coins in stacker or return the notes/coin from the dispensing outlet.		
Package Name	Cris	**Class Name**	Currency
Method Signature	int ClearJammedCurrencies(int CurrencyType, intEscrowClearanceMode, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
CurrencyType	Type of Currency to be cleared.	1	Currency Note
		2	Currency Coin
EscrowClearanceMode	If there is a currency note/coin in the escrow either it will be sent to collection bin or return from dispensing outlet.	0	Send the escrowed note(s)/coin(s) to return/dispensing outlet
		1	Send the escrowed note(s)/note(s) to the collection bin.

Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		

Return Values : An int indicating one of the following:

Value	Description
0	Note(s)/Coin(s) transported to the desired destination
1	Clearance failed due to any blocking near the dispensing outlet
2	Clearance failed due to collection bin full
18	Operation timeout occurred
20	Note Accepter not yet connected
21	Coin Accepter not yet connected
22	Coin Escrow not yet connected
28	Communication failure (Note Accepter)
29	Communication failure (Coin Accepter)
30	Communication failure (Coin Escrow)
31	Other error

4.4.15. Disconnect Device

Purpose	This API will be used to disconnect a Currency device.		
Package Name	Cris	**Class Name**	Currency
Method Signature	int DisConnectDevice(int DeviceType, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
DeviceType	Type of the Currency device to be disconnected.	1	Only Currency Note Accepter
		2	Only Currency Coin Accepter with escrow
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		

Return Values : An int indicating one of the following:

Value	Description
0	Currency Accepter disconnected successfully
1	Disconnected successfully but few notes/coins are still in escrow
18	Operation timeout occurred
20	Note Accepter not yet connected
21	Coin Accepter not yet connected
22	Coin Escrow not yet connected
28	Communication failure (Note Accepter)
29	Communication failure (Coin Accepter)
30	Communication failure (Coin Escrow)
31	Other error

4.5. SECURITY LOCK API

The methods implementing the functionalities of the Security locks and other associated devices are listed in this section.

4.5.1.Connect Device

Purpose	To connect to the Security Lock and associated devices		
Package Name	Cris	**Class Name**	Security
Method Signature	int ConnectDevice(int PortId, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
PortId	Serial Port Number to which the device is connected	0-50	If Security Lock device uses a serial port
		51	In case it uses USB port
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values			
Value	**Description**		
0	Security device connected successfully		
18	Operation timeout occurred		
20	Security device already connected		
25	Port doesn't exist		
28	Communication failure		
31	Other error		

4.5.2.Get Native Lib Version

Purpose	To get the Native Library Version		
Package Name	Cris	**Class Name**	Currency
Method Signature	String GetNativeLibVersion()		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Return Values : Version information comprising the following: **MAJOR -** Incompatible API changes **MINOR -** Functionality adding in a backwards-compatible manner **BUILD** - Bug fixing backwards-compatible			
Value	**Description**		
00.00.00	Error retreiving version information		
01.00.00	A valid version string		

4.5.3.Get Security Device FW Version

Purpose	To get the Security Device FW Version		
Package Name	Cris	**Class Name**	Security
Method Signature	String GetSecurityDevFWVersion()		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Return Values : Version information			
Value	**Description**		
00.00.00	Error retreiving version information		
01.00.00	A valid version string in the format or equivalent as received by the device		

4.5.4.Security Door Status

Purpose	This function checks the status of individual doors (depending on the DoorType parameter). The door types may be the main door or cash-box door.		
Package Name	Cris	**Class Name**	Security
Method Signature	int GetDoorStatus(int DoorType)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
DoorType	If there is a currency note/coin in the escrow either it will be sent to collection bin or return from dispensing outlet.	0	Main door
		1	Cash-box door
Return Values : An int indicating whether the door is opened or closed			
Value	**Description**		
0	Door open		
1	Door closed		
18	Operation timeout occurred		
20	Security device not yet connected		
28	Communication failure		
31	Other error		

4.5.5.Disable Alarm

Purpose	In general security mechanism will be such that whenever main door or cash box door is found open, system will sound alarm. Application disables alarm for a predefined time period based on successful user authentication when doors are opened.		
Package Name	Cris	**Class Name**	Security
Method Signature	int DisableAlarm(int DoorType, int Time)		
Input Parameters			

Parameter	**Description**	**Value/ Example**	**Meaning**
DoorType	Door type for which alarm to be disabled.	0	Both door
		1	Main door
		2	Cash-box door
Type	Time specified in seconds to disable the alarm.		

Return Values : An int indicating whether the door is opened or closed

Value	**Description**
0	Alarm disabled
1	Alarm could'nt be disabled
18	Operation timeout occurred
20	Security device not yet connected
28	Communication failure
31	Other error

4.5.6.Disconnect Device

Purpose	This API will be used to disconnect the Security Lock and associated devices.		
Package Name	Cris	**Class Name**	Security
Method Signature	int DisConnectDevice(int Timeout)		
Input Parameters			

Parameter	**Description**	**Value/ Example**	**Meaning**
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		

Return Values : An int indicating one of the following:

Value	**Description**
0	Security device disconnected successfully
18	Operation timeout occurred
20	Security device not yet connected
28	Communication failure
31	Other error

4.6. UPS API

The methods implementing the functionalities of the UPS access are listed in this section.

4.6.1.Connect Device

Purpose	To connect to the UPS device		
Package Name	Cris	**Class Name**	UPS
Method Signature	int ConnectDevice(int PortId, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
PortId	Serial Port Number to which the device is connected	0-50	If UPS device uses a serial port
		51	In case it uses USB port
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values			
Value	**Description**		
0	UPS connected successfully		
18	Operation timeout occurred		
20	UPS already connected		
25	Port doesn't exist		
28	Communication failure		
31	Other error		

4.6.2.Get Native Lib Version

Purpose	To get the Native Library Version		
Package Name	Cris	**Class Name**	UPS
Method Signature	String GetNativeLibVersion()		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Return Values : Version information comprising the following: **MAJOR -** Incompatible API changes **MINOR -** Functionality adding in a backwards-compatible manner **BUILD -** Bug fixing backwards-compatible			
Value	**Description**		
00.00.00	Error retreiving version information		
01.00.00	A valid version string		

4.6.3.Get UPS FW Version

Purpose	To get the UPS FW Version		
Package Name	Cris	**Class Name**	UPS
Method Signature	String GetUPSFWVersion()		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Return Values : Version information			
Value	**Description**		
00.00.00	Error retreiving version information		
01.00.00	A valid version string in the format or equivalent as received by the device		

4.6.4.Get UPS Status

Purpose	To check whether the main supply is on/off.		
Package Name	Cris	**Class Name**	UPS
Method Signature	int GetUPStatus()		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Return Values :			
Value	**Description**		
0	Main power supply is on		
1	Main power supply is down and the Kiosk is running on UPS battery		
18	Operation timeout occurred		
20	UPS not yet connected		
28	Communication failure		
31	Other error		

4.6.5.Get Battery status

Purpose	To check the consumed battery power of the UPS.		
Package Name	Cris	**Class Name**	UPS
Method Signature	int GetBatteryStatus()		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Return Values :			

Value	Description
-x	A -ve value from -1 to -100 indicating % level of battery power already consumed.
18	Operation timeout occurred
20	UPS not yet connected
28	Communication failure
31	Other error

4.6.6.Disconnect Device

Purpose	This API will be used to disconnect the UPS device.		
Package Name	Cris	**Class Name**	UPS
Method Signature	int DisConnectDevice(int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values : An int indicating one of the following:			
Value	**Description**		
0	UPS disconnected successfully		
18	Operation timeout occurred		
20	UPS not yet connected		
28	Communication failure		
31	Other error		

4.7. Printer API

The methods implementing the functionalities of the Receipt Printer are listed in this section.

4.7.1.Connect Device

Purpose	To connect to the Printer device		
Package Name	Cris	**Class Name**	Printer
Method Signature	int ConnectDevice(int PortId, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
PortId	Serial Port Number to which the device is connected	0-50	If Printer device uses a serial port
		51	In case it uses USB port
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values			
Value	**Description**		
0	Printer connected successfully		
18	Operation timeout occurred		
20	Printer already connected		
25	Port doesn't exist		
28	Communication failure		
31	Other error		

4.7.2.Get Native Lib Version

Purpose	To get the Native Library Version		
Package Name	Cris	**Class Name**	Printer
Method Signature	String GetNativeLibVersion()		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Return Values : Version information comprising the following: **MAJOR -** Incompatible API changes **MINOR -** Functionality adding in a backwards-compatible manner **BUILD -** Bug fixing backwards-compatible			
Value	**Description**		
00.00.00	Error retreiving version information		
01.00.00	A valid version string		

4.7.3.Get Printer FW Version

Purpose	To get the Printer FW Version		
Package Name	Cris	**Class Name**	Printer
Method Signature	String GetPrinterFWVersion()		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Return Values : Version information			
Value	**Description**		
00.00.00	Error retreiving version information		
01.00.00	A valid version string in the format or equivalent as received by the device		

4.7.4.Get Printer Status

Purpose	This API communicates the Printer device status.		
Package Name	Cris	**Class Name**	Printer
Method Signature	int GetPrinterStatus(int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values : An int indicating one of the following:			
Value	**Description**		
1	Printer Ready		
2	Printer not Ready		
4	Paper Jam		
5	Paper roll nearly empty		
6	Paper roll empty		
18	Operation timeout occurred		
20	Printer not yet connected		
28	Communication failure		
31	Other error		

4.7.5.Start Print

Purpose	This API makes the printer ready for printing.		
Package Name	Cris	**Class Name**	Printer
Method Signature	int StartPrint(int Timeout)		
Input Parameters			

Parameter	Description	Value/ Example	Meaning
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values : An int indicating one of the following:			
Value	**Description**		
0	Printed ready to print		
1	Printer not Ready to print		
2	Paper Jam		
3	Paper roll nearly empty		
4	Paper roll empty		
18	Operation timeout occurred		
20	Printer not yet connected		
28	Communication failure		
31	Other error		

4.7.6.Print Logo

Purpose	This API prints a Logo (image e.g. .png).		
Package Name	Cris	**Class Name**	Printer
Method Signature	int PrintLogo(byte [] Logo, int Alignment, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Logo	Logo data in byte array form		
Alignment	Alignment of Logo	0	Centre
		1	Left
		2	Right
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values : An int indicating one of the following:			
Value	**Description**		
0	Printed successfully		
1	Printer not Ready to print		
2	Paper Jam		
3	Paper roll nearly empty		
4	Paper roll empty		
18	Operation timeout occurred		
20	Printer not yet connected		
28	Communication failure		
31	Other error		

4.7.7.Print Text Line

Purpose	This API prints a line of text at the next line		
Package Name	Cris	**Class Name**	Printer
Method Signature	int PrintTextLine(String Text, int Alignment, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Text	Text to be printed		
Alignment	Alignment of Text	0	Centre
		1	Left
		2	Right
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values : An int indicating one of the following:			
Value	**Description**		
0	Printed successfully		
1	Printer not Ready to print		
2	Paper Jam		
3	Paper roll nearly empty		
4	Paper roll empty		
18	Operation timeout occurred		
20	Printer not yet connected		
28	Communication failure		
31	Other error		

4.7.8.Print Blank Line

Purpose	This API leaves a blank line		
Package Name	Cris	**Class Name**	Printer
Method Signature	int PrintBlankLine(int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values : An int indicating one of the following:			
Value	**Description**		
0	Printed successfully		
1	Printer not Ready to print		

2	Paper Jam
3	Paper roll nearly empty
4	Paper roll empty
18	Operation timeout occurred
20	Printer not yet connected
28	Communication failure
31	Other error

4.7.9.End Print

Purpose	This API ends the printing and cuts the paper.		
Package Name	Cris	**Class Name**	Printer
Method Signature	int EndPrint(int PaperCuttingMethod, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
PaperCuttingMethod	It determines whether the cutter of the printer leaves the printed paper half cut or full.	1	Half Cut
		2	Full Cut
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		
Return Values : An int indicating one of the following:			
Value	**Description**		
0	Successfully ends printing		
1	Printing not yet started		
2	Paper Jam		
3	Cutter error		
18	Operation timeout occurred		
20	Printer not yet connected		
28	Communication failure		
31	Other error		

4.7.10. Exchange Command

Purpose	This API transport a printing command to the printer		
Package Name	Cris	**Class Name**	Printer
Method Signature	int XChangeCommande(String Command, int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Command	Printer Command to be sent to the Printer		
Timeout	Time in milliseconds the API will try to perform its intended operation		

	otherwise return timeout status.		

Return Values : An int indicating one of the following:

Value	Description
0	Command executed successfully
1	Invalid printer command
18	Operation timeout occurred
20	Printer not yet connected
28	Communication failure
31	Other error

4.7.11. Disconnect Device

Purpose	This API will be used to disconnect the Printer device.		
Package Name	Cris	**Class Name**	Printer
Method Signature	int DisConnectDevice(int Timeout)		
Input Parameters			
Parameter	**Description**	**Value/ Example**	**Meaning**
Timeout	Time in milliseconds the API will try to perform its intended operation otherwise return timeout status.		

Return Values : An int indicating one of the following:

Value	Description
0	Printer disconnected successfully
18	Operation timeout occurred
20	Printer not yet connected
28	Communication failure
31	Other error