RFID SDK USER MANUAL

Rev1.4.6 2025-09-08

BLUEBIRD
A TSC Company

Revision History

Version	Date	Changes	Library version	Author	Approver
1.4.6	2025-09-08	Add API	5.80.27.24	JW Lee	JW Lee
1.4.5	2025-06-16	Internal Update	5.80.27.23	JW Lee	JW Lee
1.4.4	2025-04-10	Internal Update	5.80.27.22	JW Lee	JW Lee
1.4.3	2025-03-25	Internal Update	5.80.27.21	JW Lee	JW Lee
1.4.2	2025-03-04	Internal Update	5.80.27.20	JW Lee	JW Lee
1.4.1	2025-02-26	Internal Update	5.80.27.19	JW Lee	JW Lee
1.4.0	2025-01-24	Internal Update	5.80.27.18	JW Lee	JW Lee
1.3.0	2024-12-31	Internal Update	5.80.27.17	JW Lee	JW Lee
1.2.0	2024-12-09	Internal Update	5.80.27.16	JW Lee	JW Lee
1.1.9	2024-06-26	Internal Update	5.80.27.15	JW Lee	JW Lee
1.1.8	2024-03-14	Internal Update	5.80.27.14	JW Lee	JW Lee
1.1.7	2023-12-05	Internal Update	5.80.27.10	JW Lee	JW Lee
1.1.6	2023-10-13	Internal Update	5.80.27.09	JW Lee	JW Lee
1.1.5	2023-09-06	Internal Update	5.80.27.08	JW Lee	JW Lee
1.1.4	2023-06-27	Internal Update	5.80.27.07	HS Seo	HS Seo
1.1.3	2023-05-16	Internal Update	5.80.27.06	HS Seo	HS Seo
1.1.2	2023-03-15	Internal Update	5.80.27.05	JW Lee	JW Lee

Proprietary and Confidential

www.bluebirdcorp.com

For more information, contact jwlee@bluebirdcorp.com

1.1.1	2023-02-01	Internal Update	5.80.27.04	JW Lee	JW Lee
1.1.0	2022-12-20	Internal Update	5.80.27.03	JW Lee	JW Lee
1.0.9	2022-11-29	Internal Update	5.80.27.00	Hana Han	Hana Han
1.0.8	2022-10-27	Internal Update	5.80.26.00	Hana Han	Hana Han
1.0.7	2022-10-21	Internal Update	5.80.25.00	YK Jung	YK Jung
1.0.6	2022-09-15	Internal Update	5.80.24.00	YK Jung	YK Jung
1.0.5	2022-09-07	Internal Update	5.80.23.00	YK Jung	YK Jung
1.0.4	2022-08-29	Internal Update	5.80.22.00	YK Jung	YK Jung
1.0.3	2022-08-25	Internal Update	5.80.21.00	YK Jung	YK Jung
1.0.2	2022-08-18	Update API for BT Connect	5.80.20.00	YK Jung	YK Jung
1.0.1	2022-08-04	Internal Update	5.80.10.00	YK Jung	YK Jung
1.0.0	2022-07-22	Integrated Version of RFR900, RFR901	5.80.00.00	YK Jung	YK Jung

This user manual is protected by copyright.

Registered Trademark

BLUEBIRD is an emerging global brand, striving to lead the market in performance and mobility. Their products represent reliability, innovation, and innovative technology. Bluebird is a registered trademark from the global brand of Bluebird Inc. and is copyright protected.

- Bluebird and stylized Bluebird Logo are registered trademarks and symbols of Bluebird Inc.
- All other trademarks and copyrights are the property of their respective owners.

Contents

1. Introduction 12
2. Getting Started 12
1) Add Reader Protocol 13
2) Connect Reader 13
3) Disconnect Reader 13
4) Connect Sled 13
5) Disconnect Sled 13
6) Write Reader Event Handler 13
7) Run/Stop Inventory 13
8) Change Reader Configuration 13
9) Package name 14
■ Serial 14
■ Bluetooth 14
10) Abstract Reader class 14
: "Reader" and "BTReader" class inherit below abstract class 14
■ AbstractReader 14
- IBarcodeController 14
- IRfidAccess 14
- IRfidConfig 14
- IRfidInventory 14
- ISledbarcodeController 14
- ISledConfig 14
- ISledCommunicationManager 14
11) Interface class 14
■ IBarcodeController 14
■ IRfidAccess 14
■ IRfidConfig 15
■ IRfidInventory 16
■ ISledBarcodeController 16
■ ISledConfig 16
■ ISledCommunicationManager 17
■ ISerialManager 17
■ IBluetoothManager 18
12) Structure of SDK 18
13) SLED Broadcast 18
■ Broadcast Action Type 18
■ Receive way 19
① AndroidManifest.xml 19
② RFIDReceiver.java 19

RFID SDK

14) Regarding to Serial number and models 19

- RFR900 19
- RFR900 Stand alone type(attached Barcode) 20
- RFR901 20

SLED Connect / Disconnect Flow 20

3. RFID API Specification 21

1) API Class Name 21

- Serial 21
- Bluetooth 21
- Common 21

2) Constants (SDConsts class) 21

- User Open Constants 21
- Broadcast action event of SLED attached/detached : Uses only in Bluebird Android Device 24
- BB SLED callback message values : Msg class 24
- BB RF callback command message values   : RFCmdMsg class 24
- BB RF Get command common result values 25
- BB RF Access and Set command result values : RFResult class 27
- RF Dutycycle values : RFDutyCycle class 28
- RF Access timeout values : RFAccessTimeout class 28
- RF Power values : RFPower class 28
- RF Mode values : RFMode class 28
- RF Singulation values : RFSingulation class 29
- RF Region values   : RFRegion class 29
- RF RSSI values : RFRssi class 35
- RF Session values : RFSession class 35
- RF Toggle values : RFToggle class 35
- RF Dwelltime values : RFDwell class 35
- RF Inventory SessionTarget values : RFInvSessionTarget class 36
- RF Selection Flag values : RFSelectionFlag class 36
- RF Memory Type values : RFMemType class 36
- RF ISO Region values : RFISORegion class 36
- BB Sled callback command message values : SDCmdMsg class 39
- BB Sled callback Error String result value 40
- BB Sled callback Not Supported or   Not Supported yet String result value 40
- BB Sled Get command common result values 41
- BB Sled Set command result values : SDResult class 42
- SD Battery State values : SDBatteryState class 42
- SD Trigger State values(BB SLED mode values) : SDTriggerMode class 42
- SD Hotswap State values(BB SLED mode values) : SDHotswapState class 42
- SD Buzzer Level values : SDBuzzerLevel class 43
- SD Sleep Timeout values : SDSleepTimeout class 43

- SD Connect State values : SDConnectState class : Only for Serial interface (Reader) 43
- SD Buzzer State values : SDConnectState class 44
- SD Charge State values : SDChargeState class 44
- SD Mode Key Enable State values : SDModeKeyState class....44
- SD Trigger Key Enable State values : SDTriggerKeyState class 44
- SD Tag Buzzer Enable State values : SDTagBuzzerState class 45
- SD LED Enable State values : SDLEDState class....45
- SD Wakeup state values(BB SLED State values) : SDState class....45
- BB Barcode command message values : BCCmdMsg class 45
- BB Barcode common callback result values : BCCommonResult class 46
- BB Barcode callback result values : BCResult class 47
- BB Barcode HW key format value : BCKeyFormat class 47
- BB Barcode state value : BCState class 47
- BB Barcode Multi Scan State value : BCMultiScanState class 47
- BB Barcode Multi Scan Type value : BCMultiScanType class....48
- BB Barcode Trigger Mode : BCBarcodeTriggerMode class 48
- BB Barcode Multi Scan number : BCBarcodeMultiNumber class 48
- BB Barcode attached to SLED command message values : SBCmdMsg class 48
- BB Barcode attached to SLED callback result values : SBCommonResult class....49
- BB Barcode attached to SLED callback result values : SBResult class 50
- SB Preset text type values : SBPresetType class 50
- SB Barcode Trigger Mode : SBBarcodeTriggerMode class....50
- SB Param values : SBParam class....51
- BB Barcode Symbology values : SymbologyType class....72
- BB Sled BT callback command message values : BTCmdMsg class 76
- BB Sled Get command common result values : BTCommonResult class 77
- BB Sled Set command result values : BTResult class 78
- BB Sled BT State : BTState class 79
- BB Sled BT Bond State : BTBondState class....79
- BB Sled Set command result values : BTConnectState class 79
- BB Sled Bluetooth device type : BTDeviceType class 80

3) Default....80
- Default Value(ResetConfigToFactoryDefaults API)....80

4) Arguments 80
- Describe arguments of RF APIs....80

5) Event Handler....81

6) Barcode mode 81

7) Barcode parameters 82

8) Selection Criterias....86
- SelectionCriterias's Criteria Constants 86
- SelectionCriterias Memory Type : SCMemType class....86

- SelectionCriterias Action Type : SCActionType class 87
- SelectionCriterias Result : Result class 88
- SelectionCriterias constructor : SelectionCriterias() 88
- getCriteria value : getCriteria class 88
- makeCriteria API 88
- Criteria class 89

9) Global Region 89

- RFR900Wxxx /RFR901Wxxx/HF550XRWxxx(EU) 89
- RFR900Nxxx/RFR901Nxxx/HF550XRNxxx (FCC) 91
- RFR900Cxxx /RFR901Cxxx/HF550XRCxxx (CH) 92
- RFR900J1xxx/RFR901J1xxx (JP) 92
- RFR900J2xxx/RFR901J2xxx/HF550XRJ2xxx (JP) 92
- RFR900DZxxx/RFR901DZxxx/HF550XRDZxxx (Algeria) 92
- RFR900MAxxx/RFR901MAxxx/HF550XRMAxxx (Morocco) 93
- RFR900EGxxx/RFR901EGxxx/HF550XREGxxx (Egypt) 93
- RFR900CLxxx/RFR901CLxxx/HF550XRCLxxx (Chile) 93
- RFR900ILxxx /RFR901ILxxx/HF550XRILxxx (Israel) 93

10) BC Barcode Lifecycle 94

11) Bluetooth callback message (BTReader) 94

- SLED_BT_DEVICE_FOUND 94
- SLED_BT_BOND_STATE_CHANGED 94
- SLED_BT_ACL_CONNECTED 94
- SLED_BT_ACL_DISCONNECT_REQUESTED 95
- SLED_BT_ACL_DISCONNECTED 95
- SLED_BT_STATE_CHANGED 95
- SLED_BT_DISCOVERY_STARTED 95
- SLED_BT_DISCOVERY_FINISHED 95
- SLED_BT_PAIRING_REQUEST 95

12) APIs 95

Reader 95

BTReader 96

- RF APIs 97

RF_GetDutyCycle 97

RF_SetDutyCycle 97

RF_GetAccessTimeout 98

RF_SetAccessTimeout 98

RF_GetRadioPowerState 99

RF_SetRadioPowerState 100

RF_GetRFMode 101

RF_SetRFMode 101

RF_GetSingulationControl 102

RF_GetMinSingulationControl 103
RF_GetMaxSingulationControl 103
RF_SetSingulationControl 104
RF_ResetConfigToFactoryDefaults 105
RF_GetRegion 106
RF_SetRegion 106
RF_GetAvailableRegionAtThisDevice 108
~~RF_SetRegionAuto~~ 108
RF_GetLibVersion 109
~~RF_Open~~ 109
~~RF_Open~~ 109
~~RF_Close~~ 110
RF_GetRssiTrackingState 110
RF_SetRssiTrackingState 111
RF_GetSession 111
RF_SetSession 112
RF_GetToggle 113
RF_SetToggle 113
RF_RemoveSelection 114
RF_SetSelection 115
RF_GetSelection 115
RF_ModuleReboot 116
RF_GetDwelltime 116
RF_SetDwelltime 117
RF_GetRFIDVersion 118
RF_UpdateRFIDFirmware 118
RF_UpdateRFIDFirmware 119
RF_GetInventorySessionTarget 121
RF_SetInventorySessionTarget 121
RF_GetSelectionFlag 122
RF_SetSelectionFlag 123
RF_PerformInventory 123
RF_PerformInventory 124
RF_PerformInventoryWithLocating 126
RF_PerformInventoryForLocating 127
RF_PerformInventoryWithPhaseFreq 128
RF_PerformInventoryCustom 129
RF_PerformInventoryWithRssiLimitation 130
RF_PerformInventoryEncoding 131
RF_SetEncodeInformation 132
RF_StopInventoryEncoding 133

RFID SDK

RF_StopInventory 134
RF_BlockWrite 135
RF_BlockPermalock 136
RF_BlockErase 137
RF_KILL 138
RF_LOCK 139
RF_READ 141
RF_WRITE 142
RF_WriteAccessPassword 144
RF_WriteKillPassword 145
RF_WriteTagID 146
RF_BulkWrite 147
RF_WriteSwitchMode 149
RF_SetEnableChannels 150
RF_SetEnableChannels 151
RF_SetRegionISO 152
RF_checkRegionISO 152
RF_GetEnableChannels 153
RF_GetDefaultChannels 153
RF_SetLBTVaule 153
RF_GetLBTVaule 154
RF_SetDYNRFMode 155
RF_GetDYNRFMode 156
RF_SetDYNStartQ 156
RF_GetDYNStartQ 157
RF_SetDYNModeSquence 158
RF_GetDYNModeSquence 158
RF_SetDYNModeMinMaxMode 159
RF_GetDYNModeMinMaxMode 160
RF_UpdateDYNProfile 160
RF_UpdateDYNProfileFCC 161
RF_UpdateDYNProfileEU 162
RF_StartCarrierWave 163
RF_StopCarrierWave 163
RF_SetRFIDProtocolType 164
RF_GetRFIDProtocolType 165
- SD APIs 165
SD_Open 165
SD_Open 166
SD_Close 166
SD_GetVersion 166

SD_GetBootLoaderVersion 167
SD_GetBatteryStatus 168
SD_GetTriggerMode 168
SD_SetTriggerMode 169
SD_Connect 170
SD_Disconnect 170
SD_SetBuzzerLevel 170
SD_GetBuzzerLevel 171
SD_SetAutoSleepTimeout 172
SD_GetAutoSleepTimeout 172
SD_GetConnectState 173
SD_SetBuzzerEnable 173
SD_GetBuzzerState 174
SD_SetTagBuzzerEnable 175
SD_GetTagBuzzerState 176
SD_SetTagBuzzerSound 176
SD_SetLEDEnable 177
SD_GetLEDEnableState 178
SD_Wakeup 178
SD_GetChargeState 179
SD_GetSerialNumber 179
SD_GetHostSerialNumber 180
SD_UpdateSLEDFirmware 180
SD_UpdateSLEDFirmware 181
~~SD_UpdateSLEDBootloader~~ 183
SD_SmartUpdateSLEDFirmware 183
SD_SetModeKeyEnable 184
SD_GetModeKeyEnableState 185
SD_SetTriggerKeyEnable 186
SD_GetTriggerKeyEnableState 186
SD_SetBTName 187
SD_GetBTName 188
SD_GetBTVersion 188
SD_ResetConfiguration 189
SD_UpdateSLEDFirmwareAndDYN 190
SD_GetSmartBatterySerial 192
SD_GetSmartBatteryStatus 192
SD_GetSmartBatteryVoltage 193
SD_GetSmartBatteryPresentStatus 193
SD_GetSmartBatteryLevel 194
SD_GetSmartBatteryLifeTime 195

SD_GetSmartBatteryHealth 195
SD_GetSmartBatteryTemperature 196
SD_GetSmartBatteryCycleCnt 197
SD_GetSmartBatteryCapacity 197
SD_GetSmartBatteryCycleCnt 198
SD_GetType 199
- SB APIs 200
SB_ResetBarcodeConfiguration 200
SB_EnableBarcodeSound 200
SB_GetBarcodeSoundState 201
SB_SetBarcodeTriggerMode 202
SB_GetBarcodeTriggerMode 202
SB_EnableBarcodeScanner 203
SB_EnableAim 204
SB_EnableIllumination 205
SB_EnableIllumination 205
SB_GetRevision 206
~~SD_StartScanSLEDBarcode~~ 207
SB_StartScan 207
SB_GetParamValue 208
SB_SetParamValue 209
SB_SetBarcodePresetValue 209
SB_GetBarcodePresetValue 210
SB_GetSupportedDevicesInfo 211
- BC APIs 213
BC_SetTriggerState 213
BC_PauseBarcode 214
BC_ResumeBarcode 214
BC_ResumeBarcode 215
BC_GetBarcodeState 215
BC_SetBarcodeKeyFormat 216
BC_GetBarcodeKeyFormat 216
BC_SetBarcodeTriggerMode 217
BC_GetBarcodeTriggerMode 218
BC_SetBarcodeMultiScan 218
BC_GetBarcodeMultiScanState 219
BC_SetBarcodeMultiScanNumber 219
BC_GetBarcodeMultiScanNumber 220
BC_SetBarcodeMultiScanType 221
BC_GetBarcodeMultiScanType 221
BC_GetSupportedDevicesInfo 222

■ BT APIs 224
BT_Enable 224
BT_Disable 224
BT_IsEnabled 224
BT_GetPairedDevices 225
BT_StartScan 225
BT_StopScan 225
BT_Connect 225
BT_Connect 226
BT_Disconnect 227
BT_GetConnectState 227
BT_UnpairDevice 227
BT_UnpairAllDevices 228
BT_GetConnectedDeviceName 228
BT_GetConnectedDeviceAddr 228
4. Special note 229
1) Document Conventions 229

1. Introduction

Convention:

This document is for describing **Serial and Bluetooth Reader Control API** for the system integrators or developers for application program to apply "RFID Reader" of **Bluebird.Co.,ltd.**

The description of **Reader Control API** in this document is implemented by **Java language** by using **Eclipse IDE** development tool.

Model types of "RFID Reader" of **Bluebird.Co,ltd.** which is supported by **Host Library SDK** is as following.

2. Getting Started

We provide convenient **Reader Control** and Example Program for the system integrators or developers who intend to apply to "RFID Reader".
This chapter describes how to use the reader control focusing on the provided sample program.

The Example Program is described by using **Java** language in **Eclipse IDE** development environment. It can be applied rapidly in the following order.

▼ Add Reader Protocol

▼ Connect Reader

▼ Disconnect Reader

▼ Connect Sled

▼ Disconnect Sled

- ▼ Write Reader Event Handler
- ▼ Run/Stop Inventory
- ▼ Change Reader Configuration
- ▼ SLED Broadcast

1) Add Reader Protocol

[Protocol] – External Library (.jar)

Reader Protocol is provided as **External Library (.jar)** format.
To use provided **Reader Protocol(bluebird-sled.jar)**, firstly add on **Project** as follows.

Adding an **External Library (.jar)** using Eclipse
You can use a third party JAR in your application by adding it to your Eclipse project as follows:

- ▼ In the Package Explorer panel, **right-click** on your project and select **Properties**.
- ▼ Select **Java Build Path**, then the tab **Libraries**.
- ▼ Press the **Add External JARs...** button and select the **JAR** file.

2) Connect Reader

Please refer to the sample code.

3) Disconnect Reader

Please refer to the sample code.

4) Connect Sled

Please refer to the sample code.

5) Disconnect Sled

Please refer to the sample code.

6) Write Reader Event Handler

Please refer to the sample code.

7) Run/Stop Inventory

Please refer to the sample code.

8) Change Reader Configuration

Please refer to the sample code

9) Package name

- **Serial**

  co.kr.bluebird.sled.Reader

- **Bluetooth**

  co.kr.bluebird.sled.BTReader

10) Abstract Reader class

: "Reader" and "BTReader" class inherit below abstract class

- AbstractReader
  - IBarcodeController
  - IRfidAccess
  - IRfidConfig
  - IRfidInventory
  - ISledbarcodeController
  - ISledConfig
  - ISledCommunicationManager

11) Interface class

- IBarcodeController
  - Interface definition for BarcodeController (Uses only in models that attached a barcode H/W on Bluebird android device)

    : BC_setTriggerState(boolean isPress)
    : BC_PauseBarcode()
    : BC_ResumeBarcode()
    : BC_GetBarcodeState()
    : BC_SetBarcodeKeyFormat(int format)
    : BC_GetBarcodeKeyFormat:
    : BC_SetBarcodeTriggerMode(int BCBarcodeTriggerMode)
    : BC_GetBarcodeTriggerMode
    : BC_SetBarcodeMultiScan(int BCMultiScanState)
    : BC_GetBarcodeMultiScanState
    : BC_SetBarcodeMultiScanNumber(int BCBarcodeMultiNumber)
    : BC_GetBarcodeMultiScanNumber
    : BC_SetBarcodeMultiScanType(int BCMultiScanType)
    : BC_GetBarcodeMultiScanType

- IRfidAccess
  - Interface definition for RFID access.

: RF_BlockWrite(int RFMemType, int offset, String data, String accessPassword)

: RF_BlockPermalock(int blockPtr, int blockRange, int action, String accessPassword)

: RF_BlockErase(int RFMemType, int offset, int count, String accessPassworkd)

: RF_KILL(String killPassworkd, String accessPassworkd, boolean enableSelection)

: RF_LOCK(String lockMask, String action, String accessPassword, boolean enableSelection)

: RF_READ(int RFMemType, int startlocation, int length, String accessPassword, boolean enableSelection)

: RF_WRITE(int RFMemType, int startlocation, String data, String accessPassword, boolean enableSelection)

: RF_WriteAccessPassword(String data, String accessPassword, boolean enableSelection)

: RF_WriteKillPassword(String data, String accessPassword, boolean enableSelection)

: RF_WriteTagID(int startlocation, String data, String accessPassword, boolean enableSelection)

■ IRfidConfig

- Interface definition for RFID configuration

: RF_GetDutyCycle()

: RF_SetDutyCycle(int millisec)

: RF_GetAccessTimeout()

: RF_SetAccessTimeout(int millisec)

: RF_getRadioPowerState()

: RF_SetRadioPowerState(int RFPower)

: RF_GetRFMode()

: RF_SetRFMode(int RFMode)

: RF_GetSingulationControl()

: RF_GetMinSingulationControl()

: RF_GetMaxSingulationControl()

: RF_SetSingulationControl(int RFSingulation, int minSingulation, int maxSingulation)

: RF_ResetConfigToFactoryDefaults()

: RF_GetRegion()

: RF_SetRegion(int RFRegion)

: RF_GetAvailableRegionAtThisDevice()

: RF_GetLibVersion()

: RF_GetRssiTrackingState()

: RF_SetRssiTrackingState(int RFRssi)

: RF_GetSession()

: RF_SetSession(int RFSession)

: RF_GetToggle()

: RF_SetToggle(int RFToggle)

: RF_RemoveSelection()

: RF_SetSelection(SelectionCriterias selectionCriteria)

: RF_GetSelection()

: RF_ModuleReboot()

: RF_GetDwelltime()

: RF_SetDwelltime(int RFDwell)

: RF_GetRFIDVersion()

: RF_UpdateRFIDFirmware(String filepath)

: RF_UpdateRFIDFirmware(Uri uri)

: RF_GetInventorySessionTarget()

: RF_SetInventorySessionTarget(int RFInvSessionTarget)

: RF_GetSelectionFlag()

: RF_SetSelectionFlag()

IRfidInventory

- Interface definition for RFID inventory

: RF_PerformInventory(boolean turboMode, boolean enableSelection, boolean ignorePC)

: RF_PerformInventory(boolean turboMode, boolean enableSelection, boolean ignorePC, boolean isEPCDecoder)

: RF_PerformInventoryForLocating(String epc)

: RF_StopInventory()

: RF_PerformInventoryWithLocating(boolean turboMode, boolean enableSelection, boolean ignorePC)

ISledBarcodeController

- Interface definition for sled barcode controller(Uses only in models that attached a barcode H/W on Bluebird android device.)

: SB_ResetBarcodeConfiguration();

: SB_EnableBarcodeSound(boolean enable);

: SB_GetBarcodeSoundState();

: SB_SetBarcodeTriggerMode(int SBBarcodeTriggerMode);

: SB_GetBarcodeTriggerMode();

: SB_EnableBarcodeScanner(boolean enable);

: SB_EnableAim(boolean enable);

: SB_EnableIllumination(boolean enable);

: SB_EnableIllumination(boolean enable, byte[] imageData);

: SB_GetRevision();

: SB_StartScan(boolean start);

: SB_GetParamValue(int SBParam);

: SB_SetParamValue(int SBParam, int paramData);

: SB_SetBarcodePresetValue(int SBPresetType, String presetData);

: SB_GetBarcodePresetValue(int SBPresetType);

ISledConfig

- Interface definition for sled configuration

: SD_GetVersion();

: SD_GetBootLoaderVersion();

: SD_GetBatteryStatus();

: SD_GetTriggerMode();

: SD_SetTriggerMode(int SDTriggerMode);

: SD_SetBuzzerLevel(int SDBuzzerLevel);

: SD_GetBuzzerLevel();

: SD_SetAutoSleepTimeout(int SDSleepTimeout);

: SD_GetAutoSleepTimeout();

: SD_SetTagBuzzerSound();

: SD_SetTagBuzzerEnable(int SDTagBuzzerState);
: SD_GetTagBuzzerEnable();
: SD_SetBuzzerEnable(int SDBuzzerMute);
: SD_GetBuzzerState();
: SD_SetLEDEnable(int SDLEDState);
: SD_GetLEDEnableState();
: SD_GetChargeState();
: SD_GetSerialNumber();
: SD_UpdateSLEDFirmware(String filepath);
: SD_UpdateSLEDFirmware(Uri uri);
: SD_SmartUpdateSLEDFirmware(String filepath);
: SD_SetModeKeyEnable(int SDModeKeyState);
: SD_GetModeKeyEnableState();
: SD_SetTriggerKeyEnable(int SDTriggerKeyState);
: SD_GetTriggerKeyEnableState();
: SD_SetBTName(String SledBluetoothDeviceName);
: SD_GetBTName();
: SD_GetBTVersion();
: SD_ResetConfiguration();
: SD_UpdateSLEDFirmwareAndDYN();
: SD_GetSmartBatterySerial();
: SD_GetSmartBatteryStatus():
: SD_GetSmartBatteryVoltage();
: SD_GetSmartBatteryPersentStatus();
: SD_GetSmartBatteryLeve();
: SD_GetSmartBatteryLifeTime();
: SD_GetSmartBatteryHealth();
: SD_GetSmartBatteryTemperature();
: SD_GetSmartBatteryCycleCnt();
: SD_GetSmartBatteryCapacity();
: SD_GetType();

■ ISledCommunicationManager

- Interface definition for sled communication manager

: SD_Open();
: SD_Open(String clientId);
: SD_Close();

■ ISerialManager

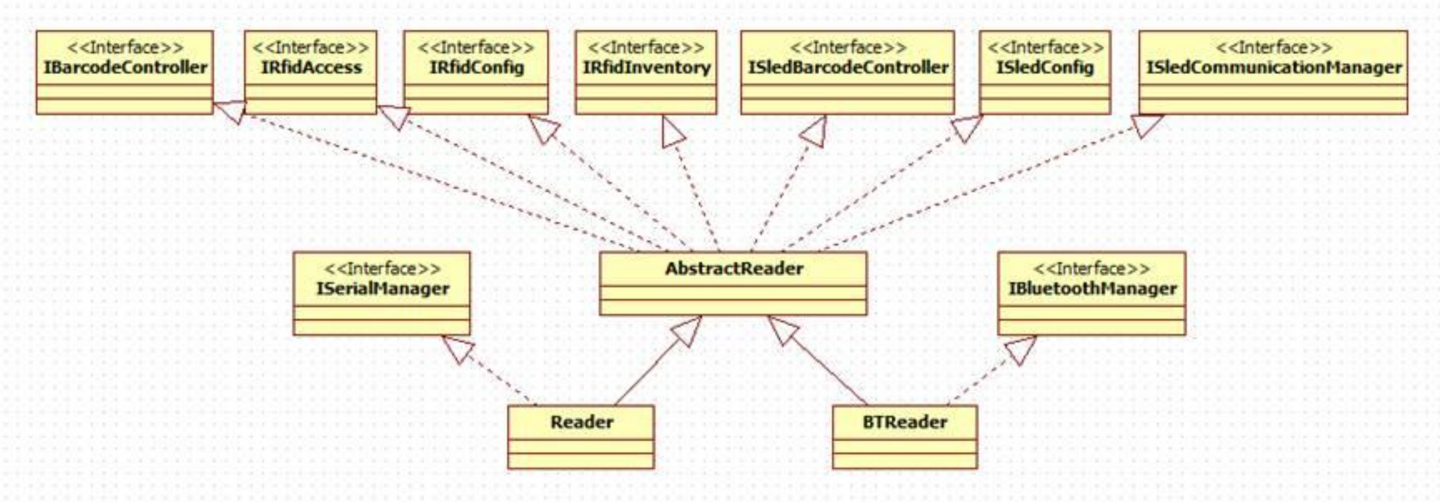
- Interface definition for serial manager(only for Serial interface(Reader))

: SD_Connect();
: SD_Disconnect();
: SD_GetConnectState();
: SD_Wakeup();

- IBluetoothManager

- Interface definition for BluetoothManager(Only for Bluetooth interface(BTReader))
  : BT_Enable();
  : BT_Disable();
  : BT_IsEnabled();
  : BT_GetPairedDevices();
  : BT_StartScan();
  : BT_StopScan();
  : BT_Connect(String address);
  : BT_Disconnect();
  : BT_GetConnectState();
  : BT_UnpairDevice(String address);
  : BT_UnpairAllDevices();
  : BT_GetConnectedDeviceName();
  : BT_GetConnectedDeviceAddr();

12) Structure of SDK



13) SLED Broadcast

Please refer to the sample code.
Uses only in Bluebird Android Device.

■ Broadcast Action Type

Type	Broadcast Action
Attached	kr.co.bluebird.android.sled.action.SLED_ATTACHED
Detached	kr.co.bluebird.android.sled.action.SLED_DETACHED

■ Receive way

① AndroidManifest.xml

Register the broadcast in 'AndroidManifest.xml' or 'source code'

```
…
<receiver android:name=".RFIDReceiver"
          android:exported="false">
          <intent-filter>
               <action android:name="kr.co.bluebird.android.sled.action.SLED_ATTACHED" />
               <action android:name="kr.co.bluebird.android.sled.action.SLED_DETACHED" />
          </intent-filter>
</receiver>
…
```

② RFIDReceiver.java

Declare broadcast action (attached/detached)

Make code with broadcast receive

```
…
    private static final String SLED_ATTACHED = "kr.co.bluebird.android.sled.action.SLED_ATTACHED";
    private static final String SLED_DETACHED = "kr.co.bluebird.android.sled.action.SLED_DETACHED";
…
    public void onReceive(Context arg0, Intent arg1) {
        if (arg1.getAction() == SLED_ATTACHED) {
            Log.d(TAG, "SLED_ATTACHED");
        }
        else if (arg1.getAction() == SLED_DETACHED) {
            Log.d(TAG, "SLED_DETACHED");
        }
    }
…
```

14) Regarding to Serial number and models

■ RFR900

– Serial Number system is as follows

Model	Region Code(Freq.)		Bluetooth available		Number
RFR900	N	KOREA, FCC and so on (900 MHz)	A	Bluetooth supported	XXXXXXXX(8digits)
	W	EU and so on (800 MHz)			
	C	CHINA			
	J1	Japan 1W			

	J2	Japan 250mW	X	Bluetooth not supported	
	DZ	Algeria			
	MA	Morocco			
	EG	Egypt			
	CL	Chile			
~~※ Japan, Algeria, Egypt, Morocco, and China should use RFR900 model only~~					

■ RFR900 Stand alone type(attached Barcode)

– Serial Number system is as follows

Model	Region Code(Freq.)		Bluetooth available		Barcode Scanner		Number
RFR900	N SN	KOREA, FCC and so on (900 MHz)	A	Bluetooth supported	S	Barcode attached	XXXXXXXX(8digits)
	W	EU and so on (800 MHz)					
	C	CHINA					
	J1	Japan 1W					
	J2	Japan 250mW	~~X~~	~~Bluetooth not supported~~			
	DZ	Algeria					
	MA	Morocco					
	EG	Egypt					
	CL	Chile					

■ RFR901

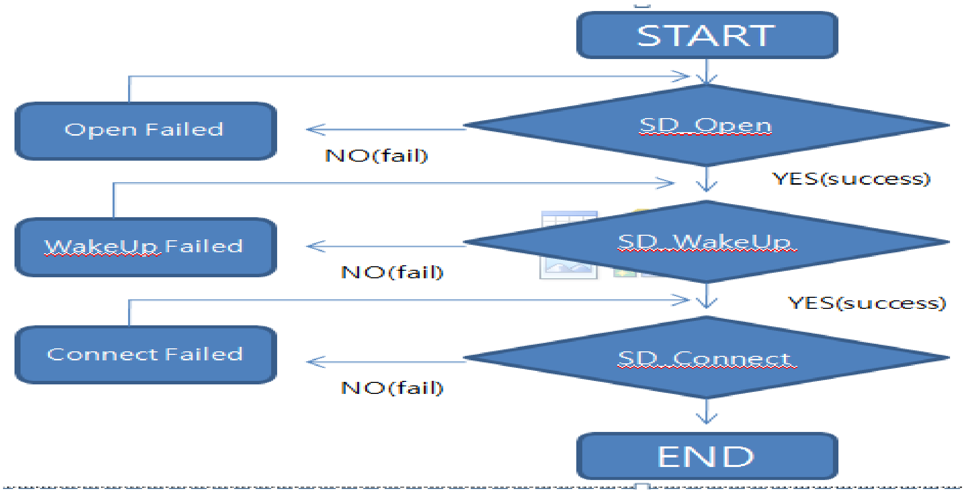
– Serial Number system is as follows

Model	Region Code(Freq.)		Bluetooth available		Number
RFR901	N	KOREA, FCC and so on (900 MHz)	A	Bluetooth supported	XXXXXXXX(8digits)
	W	EU and so on (800 MHz)			
	J1	Japan 1W			

SLED Connect / Disconnect Flow

All of the following actions have to establish if you want to connect and disconnection is performed in reverse order (wakeup(SD_WakeUp) is only used for connection).

Connection action(SD_Open/SD_WakeUp/SD_Connect) must be performed when the App is activated, and disconnection action(SD_Disconnect/SD_Close) must be performed when App is disabled.



3. RFID API Specification

1) API Class Name

- **Serial**
  - SDConsts
  - Reader
  - ISerialManager
- **Bluetooth**
  - SDConsts
  - BTReader
  - IBluetoothManager
- **Common**
  - Else

2) Constants (SDConsts class)

- **User Open Constants**

```
public static final boolean SD_OPEN_SUCCESS = true;

public static final boolean SD_OPEN_FAIL = false;

public static final int KILL_PASSWORD_LENGTH = 8;

public static final int LOCK_PASSWORD_LENGTH = 8;
```

```
    public static final int ACCESS_PASSWORD_LENGTH = 8;

    public static final int LOCK_MASK_LENGTH = 4;

    public static final int LOCK_ACTION_LENGTH = 4;

    /**
     * RSSI Prefix in Tag Data
     */
    public static final String RSSI_PREFIX_IN_TAG = ";rssi:";

    /**
     * LOCATION Prefix in Tag Data
     */
    public static final String LOCATE_PREFIX_IN_TAG = ";loc:";

    /**
     * SYMBOLOGY Prefix in Tag Data
     */
    public static final String SYM_PREFIX_IN_BARCODE = ";sym:";

    public static final int SLED_INTERFACE_ERROR = -1;

    public static final int LOW_BATTERY_PERCENT = 7;

    public static final int BARCODE_LOW_BATTERY_PERCENT = 4;


    /**
     * Only for Bluetooth interface(BTReader)
     */
    public static final String BT_BUNDLE_NAME_KEY = "name";

    /**
     * Only for Bluetooth interface(BTReader)
     */
    public static final String BT_BUNDLE_ADDR_KEY = "addr";

    /**
     * Only for Bluetooth interface(BTReader)
     */
    public static final String BT_BUNDLE_BOND_STATE_KEY = "state";

    /**
     * Only for Bluetooth interface(BTReader)
     */
    public static final String BT_BUNDLE_BOND_NEW_STATE_KEY = "new_state";

    /**
     * Only for Bluetooth interface(BTReader)
     */
    public static final String BT_BUNDLE_BOND_PREV_STATE_KEY = "pre_state";

    /**
     * Only for Bluetooth interface(BTReader)
     */
    public static final String BT_BUNDLE_CON_NEW_STATE_KEY = "new_con_state";

    /**
     * Only for Bluetooth interface(BTReader)
     */
    public static final String BT_BUNDLE_CON_PREV_STATE_KEY = "pre_con_state";

    /**
     * Only for Bluetooth interface(BTReader)
     */
    public static final int BT_NAME_MAX_LENGTH = 9;


    /**
     * Uses only in models that attached a barcode H/W on RFR900.
     */
    public static final int SB_PRESET_VALUE_MAX_LENGTH = 15;

    /**
     * Uses only in models that attached a barcode H/W on RFR900.
     */
     public static final int SB_ILLUMINATION_DATA_MAX_SIZE = 251;
```

- **Broadcast action event of SLED attached/detached**

  : Uses only in Bluebird Android Device

```
 public static final String ACTION_SLED_ATTACHED =”kr.co.bluebird.android.sled.action.SLED_ATTACHED";

public static final String ACTION_SLED_DETACHED = "kr.co.bluebird.android.sled.action.SLED_DETACHED";
```

- **BB SLED callback message values**

  : Msg class

```
    public static class Msg {
        public static final int RFMsg = 0;

        public static final int SDMsg = 1;

        /**
         * Uses only in models that attached a barcode H/W on Bluebird android device.
         */
        public static final int BCMsg = 2;

        /**
         * Only for Bluetooth interface(BTReader)
         */
        public static final int BTMsg = 3;

        /**
         * Uses only in models that attached a barcode H/W on RFR900.
         */
        public static final int SBMsg = 4;
    }
```

- **BB RF callback command message values**

  : RFCmdMsg class

```
    public static class RFCmdMsg {
        public static final int INVENTORY = 5;

        public static final int STOP_INVENTORY = 6;

        public static final int READ = 7;

        public static final int WRITE = 8;
        public static final int WRITE_ACCESS_PASSWORD = 9;

        public static final int WRITE_KILL_PASSWORD = 10;

        public static final int WRITE_TAG_ID = 11;

        public static final int BLOCK_WRITE = 12;

        public static final int BLOCK_PERMALOCK = 13;

        public static final int BLOCK_ERASE = 14;

        public static final int LOCK = 15;

        public static final int KILL = 16;

        public static final int LOCATE = 17;

        public static final int RESPONSE_CODE = 20;

        public static final int REGION_CHANGE_START = 21;

        public static final int REGION_CHANGE_END = 22;

        public static final int UPDATE_RF_FW_START = 23;

        public static final int UPDATE_RF_FW = 24;

        public static final int UPDATE_RF_FW_END = 25;

        public static final int UNKNOWN = 50;
    }
```

- **BB RF Get command common result values**

: RFCommonResult class

```
    /**
     * Uses only in models that attached a barcode H/W on RFR900
     */
    public static final int ERROR_HOTSWAP_STATE = -37;
```

```
public static final int NOT_SUPPORTED_API = -36;

public static final int ACCESS_TIMEOUT = -32;

public static final int STOP_FAILED_TRY_AGAIN = -17;

/**
 * Only for Bluetooth interface(BTReader)
 */
public static final int COMMUNICATION_ERROR = -16;

 /**
 * Only for Bluetooth interface(BTReader)
 */
public static final int BLUETOOTH_NOT_ENABLED = -15;

public static final int CHARGING_STATE_ERROR = -14;

public static final int FILE_IS_NOT_EXIST = -13;

public static final int LOW_BATTERY = -12;

public static final int NOT_INVENTORY_STATE = -11;

public static final int ALREADY_CONNECTED = -10;

public static final int ALREADY_DISCONNECTED = -9;

public static final int DUP_CMD_ERROR = -8;

public static final int READER_OR_SERIAL_STATUS_ERROR = -7;

public static final int MODE_ERROR = -6;

public static final int SD_NOT_CONNECTED = -5;

public static final int OTHER_CMD_RUNNING_ERROR = -4;

public static final int ARGUMENT_ERROR = -3;
```

```
    public static final int ALREADY_OPENED = -2;

    public static final int OTHER_ERROR = -1;
```

- **BB RF Access and Set command result values**

  : RFResult class

```
    public static class RFResult extends RFCommonResult {
        public static final int SUCCESS = 0;

        public static final int HANDLE_MISMATCH_ERROR = 1;

        public static final int UNDEFINED = 2;

        public static final int MEMORY_OVERRUN = 3;

        public static final int MEMORY_LOCKED = 4;

        public static final int ZERO_KILL_PASSWORD = 5;

        public static final int TAG_LOST = 6;

        public static final int COMMAND_FORMAT_ERROR = 7;

        public static final int READ_COUNT_INVALID = 8;

        public static final int OUT_OF_RETRIES = 9;

        public static final int OPERATION_FAILED = 10;

        public static final int INSUFFICIENT_POWER = 11;

        public static final int CRC_ERROR_ON_TAG_RESPONSE = 12;

        public static final int NO_TAG_REPLY = 13;

        public static final int INVALID_PASSWORD = 14;

        public static final int NONSPECIFIC_ERROR = 15;
    }
```

■ RF Dutycycle values

: RFDutyCycle class

```
public static class RFDutyCycle extends RFCommonResult {
    public static final int MIN_DUTY = 0;

    public static final int MAX_DUTY = 1000;
}
```

■ RF Access timeout values

: RFAccessTimeout class

```
public static class RFAccessTimeout extends RFCommonResult {
    public static final int MIN_ACCESS_TIMEOUT = 100;

    public static final int DEFAULT_ACCESS_TIMEOUT = 3000;

    public static final int MAX_ACCESS_TIMEOUT = 10000;
}
```

■ RF Power values

: RFPower class

```
public static class RFPower extends RFCommonResult {
    public static final int MIN_POWER = 5;

    public static final int MAX_POWER = 30;
}
```

■ RF Mode values

: RFMode class

```
public static class RFMode extends RFCommonResult {
      /**
       * DSB_ASK_1(0) : R2T modulation = DSB, Tari = 25000, X = 1, PW = 12500, RTCal = 75000,
       * TRCal = 200000, DR = 2(0), Miller Number = FM0,
       * TRLink Freq = 40000, Var T2 Delay = 51, Rx Delay = 577, Min To T2 Delay = 75, Tx Prop Delay = 24
       */
    public static final int DSB_ASK_1 = 0;

    /**
     * R2T modulation = PR_ASK, Tari = 25000, X = 0, PW =   8250, RTCal = 62500,
     * TRCal =   85333, DR = 3(1), Miller Number = M4,
         * TRLink Freq = 250000, Var T2 Delay = 0, Rx Delay = 337, Min To T2 Delay = 12, Tx Prop Delay = 14
         */
        public static final int PR_ASK_1 = 1;

        /**
         * PR_ASK_2(2) : R2T modulation = PR_ASK, Tari = 25000, X = 0, PW =   8250, RTCal = 62500,
         * TRCal =   71111, DR = 3(1), Miller Number = M4,
         * TRLink Freq = 300000, Var T2 Delay = 0, Rx Delay = 337, Min To T2 Delay = 10, Tx Prop Delay = 14
         */
        public static final int PR_ASK_2 = 2;

        /**
         * DSB_ASK_2(3) : R2T modulation = DSB, Tari = 6250, X = 0, PW = 3125, RTCal = 15625,
         * TRCal = 20000, DR = 2(0), Miller Number = FM0,
         * TRLink Freq = 400000, Var T2 Delay = 0, Rx Delay = 313, Min To T2 Delay = 8, Tx Prop Delay = 7
         */
        public static final int DSB_ASK_2 = 3;
    }
```

- **RF Singulation values**

  : RFSingulation class

```
    public static class RFSingulation extends RFCommonResult {
        public static final int MIN_SINGULATION = 0;

        public static final int MAX_SINGULATION = 15;
    }
```

- **RF Region values**

  : RFRegion class

```
    public static class RFRegion extends RFCommonResult {
        public static final int UNKNOWN = -1;

        /**
         * Available only on RFR900N model.
         */
        public static final int KOREA = 0;

        /**
         * Available only on RFR900W model.
         */
        public static final int ETSI = 1;

        /**
         * Available only on RFR900N model.
         */
        public static final int FCC = 2;

        /**
         * Available only on RFR900N model.
         */
        public static final int AUSTRALIA = 3;

        /**
         * Available only on RFR900N model.
         */
        public static final int BANGLADESH = 4;

        /**
         * Available only on RFR900N model.
         */
        public static final int BRAZIL = 5;

        /**
         * Available only on RFR900N model.
         */
        public static final int BRUNEI = 6;

        /**
         * Available only on RFR900C model.
         */
        public static final int CHINA = 7;

        /**
         * Available only on RFR900N model.
         */
        public static final int HONGKONG = 8;

        /**
         * Available only on RFR900W model.
         */
        public static final int INDIA = 9;
```

```
        /**
         * Available only on RFR900N model.
         */
        public static final int INDONESIA = 10;

        /**
         * Available only on RFR900W model.
         */
        public static final int IRAN = 11;

        /**
         * Available only on RFR900N model.
         */
        public static final int ISRAEL = 12;

        /**
         * Available only on RFR900J1 model.
         */
        public static final int JAPAN_1 = 13;

        /**
         * Available only on RFR900J2 model.
         */
        public static final int JAPAN_2 = 14;

        /**
         * Available only on RFR900W model.
         */
        public static final int JORDAN = 15;

        /**
         * Available only on RFR900N model.
         */
        public static final int MALAYSIA = 16;

        /**
         * Available only on RFR900MA model.
         * Power limitation 26 dBm(500mW)
         */
        public static final int MOROCCO = 17;
        /**
         * Available only on RFR900N model.
         */
        public static final int NEW_ZEALAND = 18;

        /**
         * Available only on RFR900W model.
         */
        public static final int PAKISTAN = 19;

        /**
         * Available only on RFR900N model.
         */
        public static final int PERU = 20;

        /**
         * Available only on RFR900N model.
         */
        public static final int PHILIPPINES = 21;

        /**
         * Available only on RFR900N model.
         */
        public static final int SINGAPORE = 22;

        /**
         * Available only on RFR900N model.
         */
        public static final int SOUTH_AFRICA = 23;

        /**
         * Available only on RFR900N model.
         */
        public static final int TAIWAN = 24;

        /**
         * Available only on RFR900N model.
         */
        public static final int THAILAND = 25;
```

```
/**
 * Available only on RFR900N model.
 */
public static final int URUGUAY = 26;

/**
 * Available only on RFR900N model.
 */
public static final int VENEZUELA = 27;

/**
 * Available only on RFR900N model.
  * Power limitation   26 dBm(100mW)
 */
public static final int VIETNAM = 28;

/**
 * Available only on RFR900W model.
 */
public static final int RUSSIA = 29;

/**
 * Available only on RFR900DZ model.
 * Power limitation 17 dBm(100mW)
 */
public static final int ALGERIA = 30;

/**
 * Available only on RFR900EG model.
 * Power limitation 17 dBm(100mW)
 */
public static final int EGYPT = 31;

/**
 * Available only on RFR900CL model.
 * Power limitation 17 dBm(100mW)
 */
public static final int CHILE = 32;

/**
 *Available only on RFR900N model.
```

```
        * Power limitation 20 dBm(200mW(23dBm)-3dBm(Antenna Gain))
         */
        public static final int GUATEMALA = 33;

         *Available only on RFR900NA model.
         * Power limitation 27 dBm(1000mW(30dBm)-3dBm(Antenna Gain))
         */
         public static final int MACAO = 34;

         /**
         *Available only on RFR900NA model.
         * Power limitation 14 dBm(50mW(17dBm)-3dBm(Antenna Gain))
         */
         public static final int NICARAGUA = 35;

         /**
         *Available only on RFR900W model.
         * Power limitation 26 dBm(500mW)
         */
         public static final int CAMBODIA = 36;

         /**
         *Available only on RFR900W model.
         * Power limitation 26 dBm(500mW)
         */
         public static final int MYANMAR = 37;

         public static final int ETSI_UPPER = 38;

         /**
         *Available only on RFR900N model.
         */
         public static final int PARAGUAY = 39;

        /**
         * Region not setted state
         * In case of this state, device may not working well
         * So, use device after set region exactly
         */
         public static final int NOT_SETTED = 250;
    }
```

■ RF RSSI values

: RFRssi class

```
public static class RFRssi extends RFCommonResult {
    public static final int OFF = 0;

    public static final int ON = 1;
}
```

■ RF Session values

: RFSession class

```
public static class RFSession extends RFCommonResult {
    public static final int SESSION_S0 = 0;

    public static final int SESSION_S1 = 1;

    public static final int SESSION_S2 = 2;

    public static final int SESSION_S3 = 3;
}
```

■ RF Toggle values

: RFToggle class

```
public static class RFToggle extends RFCommonResult {
    public static final int OFF = 0;

    public static final int ON = 1;
}
```

■ RF Dwelltime values

: RFDwell class

```
public static class RFDwell extends RFCommonResult {
    public static final int MIN_DWELL = 50;

    public static final int MAX_DWELL = 400;
}
```

■ RF Inventory SessionTarget values

: RFInvSessionTarget class

```
public static class RFInvSessionTarget extends RFCommonResult {
    public static final int TARGET_A = 0;

    public static final int TARGET_B = 1;
}
```

■ RF Selection Flag values

: RFSelectionFlag class

```
public static class RFSelectionFlag extends RFCommonResult {
    public static final int ALL = 1;

    public static final int DEASSERTED = 2;

    public static final int ASSERTED = 3;
}
```

■ RF Memory Type values

: RFMemType class

```
public static class RFMemType {
    public static final int RESERVED = 0;

    public static final int EPC = 1;

    public static final int TID = 2;

    public static final int USER = 3;
}
```

■ RF ISO Region values

: RFISORegion class

```
public static class RFISORegion extends RFCommonResult {
            public static final String AE = "AE";//             U.A.E. :EU
            public static final String AM = "AM";//             Armenia :EU
            public static final String AT = "AT";//             Austria :EU
            public static final String AZ = "AZ";//             Azerbaijan :EU
            public static final String BE = "BE";//             Belgium :EU
            public static final String BG = "BG";//             Bulgaria :EU
            public static final String BA = "BA";//         Bosnia :EU
            public static final String BY = "BY";//         Belarus :EU
            public static final String CH = "CH";//         Switzerland :EU
            public static final String CY = "CY";//         Cyprus :EU
            public static final String CZ = "CZ";//         Czech Republic :EU
            public static final String DE = "DE";//         Germany :EU
            public static final String DK = "DK";//         Denmark :EU
            public static final String ES = "ES";//         Spain :EU
            public static final String EE = "EE";//         Estonia :EU
            public static final String FI = "FI";//         Finland :EU
            public static final String FR = "FR";//         France :EU
            public static final String GB = "GB";//         United Kingdom :EU
            public static final String GR = "GR";//         Greece :EU
            public static final String HR = "HR";//         Croatia :EU
            public static final String HU = "HU";//         Hungary :EU
            public static final String IE = "IE";//         Ireland :EU
            public static final String IS = "IS";//         Iceland :EU
            public static final String IT = "IT";//         Italy :EU
            public static final String LT = "LT";//         Lithuania :EU
            public static final String LU = "LU";//         Luxembourg :EU
            public static final String LV = "LV";//         Latvia :EU
            public static final String MK = "MK";//         Macedonia (FYROM) :EU
            public static final String MT = "MT";//         Malta :EU
            public static final String MD = "MD";//         Moldova :EU
            public static final String NL = "NL";//         Netherlands :EU
            public static final String NO = "NO";//         Norway :EU
            public static final String NG = "NG";//         Nigeria :EU
            public static final String OM = "OM";//         Oman :EU
            public static final String PL = "PL";//         Poland :EU
            public static final String PT = "PT";//         Portugal :EU
            public static final String RO = "RO";//         Romania :EU
            public static final String RS = "RS";//         Serbia :EU
            public static final String SA = "SA";//         Saudi Arabia :EU
            public static final String SK = "SK";//         Slovakia :EU
            public static final String SI = "SI";//         Slovenia :EU
            public static final String SE = "SE";//         Sweden :EU
            public static final String TN = "TN";//         Tunisia :EU
            public static final String TR = "TR";//         Turkey :EU
            public static final String UA = "UA";//         Ukraine:EU
            public static final String AD = "AD";//         Andorra:EU
            public static final String AL = "AL";//         Albania:EU
            public static final String BH = "BH";//              Bahrain:EU
            public static final String GE = "GE";//              Georgia:EU
            public static final String KW = "KW";//              Kuwait:EU
            public static final String KZ = "KZ";//               Kazakhstan:EU
            public static final String LB = "LB";//              Lebanon:EU
            public static final String MC = "MC";//              Monaco:EU
            public static final String ME = "ME";//               Montenegro:EU
            public static final String QA = "QA";//             Qatar:EU
            public static final String XK = "XK";//             kosovo:EU
            public static final String IN = "IN";//              India :EU
            public static final String IR = "IR";//              Iran :EU
            public static final String JO = "JO";//              Jordan :EU
            public static final String PK = "PK";//              Pakistan :EU
            public static final String RU = "RU";//              Russia :EU
            public static final String KM = "KM";//              Cambodia :EU
            public static final String MM = "MM";//              Myanmar :EU
            public static final String IQ = "IQ";//              Iraq :EU

            public static final String PY = "PY";//               Paraguay:FCC
            public static final String AR = "AR";//              Argentina :FCC
            public static final String CA = "CA";//              Canada :FCC
            public static final String CO = "CO";//             Colombia :FCC
            public static final String CR = "CR";//              Costa Rica :FCC
            public static final String DO = "DO";//              Dominican Republic :FCC
            public static final String MX = "MX";//              Mexico :FCC
            public static final String PA = "PA";//              Panama :FCC
            public static final String US = "US";//              United States :FCC
            public static final String GT = "GT";//              Guatemala, 20dbm :FCC
            public static final String NI = "NI";//              Nicaragua, 14dbm :FCC
            public static final String HN = "HN";//              Honduras :FCC
            public static final String SV = "SV";//              El Salvador :FCC

            public static final String AU = "AU";//              Australia:FCC
            public static final String BD = "BD";//              Bangladesh:FCC
            public static final String BR = "BR";//              Brazil:FCC
            public static final String BN = "BN";//              Brunei Darussalam(Brunei):FCC
            public static final String ID = "ID";//              Indonesia:FCC
            public static final String HK = "HK";//              Hongkong:FCC
            public static final String SG = "SG";//              Singapore:FCC
            public static final String TH = "TH";//              Thailand:FCC
            public static final String VN = "VN";//              Vietnam:FCC
            public static final String IL = "IL";//              Israel:FCC
            public static final String KR = "KR";//              Korea:FCC
            public static final String MY = "MY";//              Malaysia:FCC
            public static final String NZ = "NZ";//              New Zealand:FCC
            public static final String PE = "PE";//              Peru:FCC
            public static final String PH = "PH";//              Philippines(Philippines):FCC
            public static final String ZA = "ZA";//              South Africa:FCC
            public static final String UY = "UY";//              Uruguay:FCC
            public static final String TW = "TW";//              Taiwan:FCC
            public static final String VE = "VE";//              Venezuela(Venezuela):FCC

            //RFR900Cxxx (CH)
            public static final String CN = "CN";//               People's Republic of China(China)

            //RFR900J1xxx (JP)
            //RFR900J2xxx (JP)
            public static final String JP = "JP";//               Japan

            //RFR900DZxxx (Algeria)
            public static final String DZ = "DZ";//                Algeria

            //RFR900MAxxx (Morocco)
            public static final String MA = "MA";//                Morocco

            //RFR900EGxxx (Egypt)
            public static final String EG = "EG";//                Egypt

            //RFR900CLxxx (Chile)
            public static final String CL = "CL";//               Chile

            public static final String MO = "MO";//               Macao
            }
```

- **BB Sled callback command message values**

  : SDCmdMsg class

※ SLED_INVENTORY_STATE_CHANGED

If error is occurred in RFR900's firmware during the inventory, this message can receive.

Notifications battery gauge every 3 seconds.

```
public static class SDCmdMsg {
    public static final int TRIGGER_PRESSED = 41;

    public static final int TRIGGER_RELEASED = 42;

    public static final int SLED_BATTERY_STATE_CHANGED = 43;

    public static final int SLED_MODE_CHANGED = 45;

    public static final int SLED_INVENTORY_STATE_CHANGED = 46;

    public static final int SLED_WAKEUP = 47;

    public static final int UPDATE_SD_FW_START = 48;

    public static final int UPDATE_SD_FW = 49;

    public static final int UPDATE_SD_FW_END = 50;

    public static final int UPDATE_SD_BOOT_START = 66;

    public static final int UPDATE_SD_BOOT = 67;

    public static final int UPDATE_SD_BOOT_END = 68;

    public static final int SLED_UNKNOWN_DISCONNECTED = 51;

    /**
     * Uses only in models that attached a barcode H/W on RFR900
     */
    public static final int SLED_HOTSWAP_STATE_CHANGED = 69;
}
```

- **BB Sled callback Error String result value**

```
public static final String ERROR_STR = "Error";
```

- **BB Sled callback Not Supported or Not Supported yet String result value**

```
public static final String NOT_SUPPORTED_API_STR = "Not Supported API";
```

■ BB Sled Get command common result values

: SDCommonResult class

```
/**
 * Uses only in models that attached a barcode H/W on RFR900
 */
public static final int ERROR_HOTSWAP_STATE = -37;

public static final int NOT_SUPPORTED_API = -36;

public static final int ACCESS_TIMEOUT = -32;

/**
 * Only for Bluetooth interface(BTReader)
 */
public static final int BT_NAME_LENGTH_ERROR = -19;

/**
 * Only for Bluetooth interface(BTReader)
 */
 public static final int COMMUNICATION_ERROR = -16;

/**
 * Only for Bluetooth interface(BTReader)
 */
public static final int BLUETOOTH_NOT_ENABLED = -15;

public static final int CHARGING_STATE_ERROR = -14;

public static final int FILE_IS_NOT_EXIST = -13;

public static final int LOW_BATTERY = -12;

public static final int ALREADY_CONNECTED = -10;

public static final int ALREADY_DISCONNECTED = -9;

public static final int DUP_CMD_ERROR = -8;

public static final int READER_OR_SERIAL_STATUS_ERROR = -7;
public static final int MODE_ERROR = -6;

public static final int SD_NOT_CONNECTED = -5;

public static final int OTHER_CMD_RUNNING_ERROR = -4;

public static final int ARGUMENT_ERROR = -3;

public static final int OTHER_ERROR = -1;
```

■ BB Sled Set command result values

: SDResult class

```
public static class SDResult extends SDCommonResult {
    public static final int SUCCESS   = 0;
}
```

■ SD Battery State values

: SDBatteryState class

```
public static class SDBatteryState extends SDCommonResult {
    public static final int MIN = 0;

    public static final int MAX = 100;
}
```

■ SD Trigger State values(BB SLED mode values)

: SDTriggerMode class

```
public static class SDTriggerMode extends SDCommonResult {
    public static final int RFID = 0;

    public static final int BARCODE = 1;
}
```

■ SD Hotswap State values(BB SLED mode values)

: SDHotswapState class

```
public static class SDHotswapState extends SDCommonResult {
    public static final int HOTSWAP_STATE = 0;

    public static final int NORMAL_STATE = 1;
    }

```

SD Buzzer Level values

: SDBuzzerLevel class

```
    public static class SDBuzzerLevel extends SDCommonResult {
        public static final int LOW = 0;

        public static final int MID = 1;

        public static final int HIGH = 2;
    }
```

SD Sleep Timeout values

: SDSleepTimeout class

```
    public static class SDSleepTimeout extends SDCommonResult {
        public static final int NO_SLEEP = 0;

        public static final int SEC_15 = 1;

        public static final int SEC_30 = 2;

        public static final int MIN_1 = 3;

        public static final int MIN_3 = 4;

        public static final int MIN_5 = 5;

        public static final int MIN_10 = 6;
    }
```

SD Connect State values

: SDConnectState class

: Only for Serial interface (Reader)

```
    public static class SDConnectState extends SDCommonResult {
        public static final int DISCONNECTED = 0;

        public static final int CONNECTED = 1;
    }
```

SD Buzzer State values

: SDConnectState class

```
    public static class SDBuzzerState extends SDCommonResult {
        public static final int MUTE = 0;

        public static final int NOISY = 1;
    }
```

SD Charge State values

: SDChargeState class

```
    public static class SDChargeState extends SDCommonResult {
        public static final int OFF = 0;

        public static final int ON = 1;

        ~~public static final int HOTSWAP = 2;~~
}
```

SD Mode Key Enable State values

: SDModeKeyState class

```
    public static class SDModeKeyState extends SDCommonResult {
        public static final int DISABLE = 0;

        public static final int ENABLE = 1;
    }
```

SD Trigger Key Enable State values

: SDTriggerKeyState class

```
    public static class SDTriggerKeyState extends SDCommonResult {
        public static final int DISABLE = 0;

        public static final int ENABLE = 1;
    }
```

■ SD Tag Buzzer Enable State values

: SDTagBuzzerState class

```
public static class SDTagBuzzerState extends SDCommonResult {
    public static final int DISABLE = 0;

    public static final int ENABLE = 1;
}
```

■ SD LED Enable State values

: SDLEDState class

```
public static class SDLEDState extends SDCommonResult {
    public static final int DISABLE = 0;

    public static final int ENABLE = 1;
}
```

■ SD Wakeup state values(BB SLED State values)

: SDState class

```
public static class SDState extends SDCommonResult {
    public static final int SLEEP = 0;

    public static final int WAKEUP = 1;
}
```

■ BB Barcode command message values

: BCCmdMsg class

※ BARCODE_READ (Different to BARCODE_READ in SDCmdMsg)
Message for barcode reading from Bluebird Android Device

※ Occurs only in models that do not have a barcode on SLED

※ Uses only in models that attached a barcode H/W on Bluebird android device

```
public static class BCCmdMsg {
    public static final int BARCODE_READ = 80;

    public static final int BARCODE_TRIGGER_PRESSED = 81;

    public static final int BARCODE_TRIGGER_RELEASED = 82;

    public static final int BARCODE_ACCESS_TIMEOUT = 84;
```

```
        public static final int BARCODE_ERROR = 85;
    }
```

- **BB Barcode common callback result values**

: BCCommonResult class

※ Uses only in models that attached a barcode H/W on Bluebird android device

```
    public static class BCCommonResult {
        /**
         * Uses only in models that attached a barcode H/W on RFR900
         */
        public static final int ERROR_HOTSWAP_STATE = -37;

        public static final int NOT_SUPPORTED_API = -36;

        public static final int BARCODE_NOT_ACTIVE = -35;

        public static final int ALREADY_PAUSE = -34;

        public static final int ALREADY_RESUME = -33;

        public static final int ACCESS_TIMEOUT = -32;

        public static final int LOW_BATTERY = -12;

        public static final int READER_OR_COM_INTERFACE_STATUS_ERROR = -7;

        public static final int MODE_ERROR = -6;

        public static final int SD_NOT_CONNECTED = -5;

        public static final int OTHER_CMD_RUNNING_ERROR = -4;

        public static final int ARGUMENT_ERROR = -3;

        public static final int OTHER_ERROR = -1;
    }
```

■ BB Barcode callback result values

: BCResult class

※ Uses only in models that attached a barcode H/W on Bluebird android device

```
public static class BCResult extends BCCommonResult {
        /**
         * Uses only in models that attached a barcode H/W on RFR900
         */
        public static final int SUCCESS = 0;
}
```

■ BB Barcode HW key format value

: BCKeyFormat class

※ Uses only in models that attached a barcode H/W on Bluebird android device

```
public static class BCKeyFormat extends BCCommonResult {
        public static final int PTT_SCAN = 0;

        public static final int SCAN_PTT = 1;

        public static final int PTT_PTT = 2;

        public static final int SCAN_SCAN = 3;
}
```

■ BB Barcode state value

: BCState class

※ Uses only in models that attached a barcode H/W on Bluebird android device

```
public static class BCState {
        public static final int ACTIVE = 0;

        public static final int PAUSED = 1;

        public static final int NOT_ACTIVE = 2;
}
```

■ BB Barcode Multi Scan State value

: BCMultiScanState class

※ Uses only in models that attached a barcode H/W on Bluebird android device

```
public static class BCMultiScanState extends BCCommonResult {
        public static final int DISABLE = 0;

        public static final int ENABLE = 1;
    }
```

■ BB Barcode Multi Scan Type value

: BCMultiScanType class

※ Uses only in models that attached a barcode H/W on Bluebird android device

```
    public static class BCMultiScanType extends BCCommonResult {
        public static final int DISABLE = 0;

        public static final int ENABLE = 1;
    }
```

■ BB Barcode Trigger Mode

: BCBarcodeTriggerMode class

※ Uses only in models that attached a barcode H/W on Bluebird android device

```
    public static class BCBarcodeTriggerMode extends BCCommonResult {
        public static final int LEVEL = 0;

        public static final int PULSE = 1;

        public static final int EDGE = 2;

        public static final int AUTOSTAND = 3;
    }
```

■ BB Barcode Multi Scan number

: BCBarcodeMultiNumber class

※ Uses only in models that attached a barcode H/W on Bluebird android device

```
    public static class BCBarcodeMultiNumber extends BCCommonResult {
        public static final int MIN = 0;

        public static final int MAX = 10;
    }
```

■ BB Barcode attached to SLED command message values

: SBCmdMsg class

※ Uses only in models that attached a barcode H/W on Bluebird android device

```
    public static class SBCmdMsg {
        public static final int BARCODE_TRIGGER_PRESSED_SLED = 86;

        public static final int BARCODE_TRIGGER_RELEASED_SLED = 87;

        public static final int BARCODE_READ = 88;

        public static final int BARCODE_RESET_CONFIG_START = 89;

        public static final int BARCODE_RESET_CONFIG_END = 90;
    }
```

- **BB Barcode attached to SLED callback result values**

  : SBCommonResult class

※ Uses only in models that attached a barcode H/W on Bluebird android device

```
    public static class SBCommonResult {
        /**
         * Uses only in models that attached a barcode H/W on RFR900
         */
        public static final int ERROR_HOTSWAP_STATE = -37;

        public static final int NOT_SUPPORTED_API = -36;

        public static final int BARCODE_NOT_ACTIVE = -35;

        public static final int ALREADY_PAUSE = -34;

        public static final int ALREADY_RESUME = -33;

        public static final int ACCESS_TIMEOUT = -32;

        /**
         * Only for Bluetooth interface(BTReader)
         */
        public static final int BLUETOOTH_NOT_ENABLED = -15;

        public static final int LOW_BATTERY = -12;

        public static final int READER_OR_COM_INTERFACE_STATUS_ERROR = -7;
        public static final int MODE_ERROR = -6;

        public static final int SD_NOT_CONNECTED = -5;

        public static final int OTHER_CMD_RUNNING_ERROR = -4;

        public static final int ARGUMENT_ERROR = -3;

        public static final int OTHER_ERROR = -1;
    }
```

■ BB Barcode attached to SLED callback result values

: SBResult class

※ Uses only in models that attached a barcode H/W on Bluebird android device

```
    public static class SBResult extends SBCommonResult {
        public static final int SUCCESS = 0;
    }
```

■ SB Preset text type values

: SBPresetType class

※ Uses only in models that attached a barcode H/W on Bluebird android device

```
   public static class SBPresetType extends SBCommonResult {
        public static final int PREFIX = 0;

        public static final int SUFFIX = 1;

        public static final int PREAMBLE = 2;

        public static final int POSTAMBLE = 3;
    }
```

■ SB Barcode Trigger Mode

: SBBarcodeTriggerMode class

※ Uses only in models that attached a barcode H/W on Bluebird android device

```
   public static class SBBarcodeTriggerMode extends SBCommonResult {
        public static final int LEVEL = 0;

        public static final int PULSE = 1;
        public static final int EDGE = 2;

        public static final int AUTOSTAND = 3;
    }
```

SB Param values

: SBParam class

※ Uses only in models that attached a barcode H/W on Bluebird android device

```
    public static class SBParam {
        /**
         * UPC-A
         * default = enable
         * range = 0:disable, 1:enable
         */
        public static final int UPC_A = 0x0001;

        /**
         * UPC-E
         * default = enable
         * range = 0:disable, 1:enable
         */
        public static final int UPC_E = 0x0002

        /**
         * UPC-E1
         * default = enable
         * range = 0:disable, 1:enable
         */
        public static final int UPC_E1 = 0x000C

        /**
         * EAN-8/JAN-8
         * default = enable
         * range = 0:disable, 1:enable
         */
        public static final int EAN8 = 0x0004

        /**
         * EAN-13/JAN-13
         * default = enable
```

```
     * range = 0:disable, 1:enable
     */
    public static final int EAN13 = 0x0003

    /**
     * Bookland EAN
     * default = enable
     * range = 0:disable, 1:enable
     */
    public static final int BOOKLAND_EAN = 0x0053

    /**
     * Bookland ISBN Format
     * default = Bookland ISBN-10
     * range = 0:Bookland ISBN-10, 1:Bookland ISBN-13
     */
    public static final int BOOKLAND_ISBN_FORMAT = 0xF140

    /**
     * Decode UPC/EAN/JAN Supplementals
     * default = Ignore Supplemental
     * range = 0:Ignore Supplemental,
     *             1:Decode UPC/EAN/JAN Only With Supplementals,
     *             2:Autodiscriminate UPC/EAN/JAN Supplementals,
     *             3:Enable Smart Supplemental Mode,
     *             4:Enable 378/379 Supplemental Mode,
     *             5:Enable 978/979 Supplemental Mode,
     *             6:Enable 414/419/434/439 Supplemental Mode,
     *             7:Enable 977 Supplemental Mode,
     *             8:Enable 491 Supplemental Mode,
     *             9:Supplemental User-Programmable Type 1,
     *             A:Supplemental User-Programmable Type 1 and 2,
     *             B:Smart Supplemental Plus User-Programmable 1,
     *             C:Smart Supplemental Plus User-Programmable 1 and 2
     */
    public static final int DECODE_UPC_EAN_SUPPLEMENTAL = 0x0010;

    /**
     * UPC/EAN/JAN Supplemental Redundancy
     * default : 10
     * range : 2 ~ 16
     * If you selected Autodiscriminate UPC/EAN/JAN Supplementals,
     * this option adjusts the number of times to decode a symbol without supplementals before transmission.
     * The range is from two to 16 times. Five or above is recommended when decoding a mix of
     * UPC/EAN/JAN symbols with and without supplementals.
     */
    public static final int UPC_EAN_SUPPLEMENTAL_REDUNDANCY = 0x0050;

    /**
     * UPC/EAN/JAN Supplemental AIM ID Format
     * default : Combined
     * range : 0:separate, 1:combined, 2:separate transmissions
     * Separate - transmit UPC/EAN with supplementals with separate AIM IDs but one transmission,
     * i.e.:
     * ]E<0 or 4><data>]E<1 or 2>[supplemental data]
     * Combined - transmit UPC/EAN with supplementals with one AIM ID and one transmission,
     * i.e.:
     * ]E3<data+supplemental data>
     * Separate Transmissions - transmit UPC/EAN with supplementals with separate AIM IDs and separate
     * transmissions,
     * i.e.:
     * ]E<0 or 4><data>
     * ]E<1 or 2>[supplemental data]
     */
    public static final int DECODE_UPC_EAN_SUPPLEMENTAL_AIM_ID = 0xF1A0;

    /**
     * Transmit UPC-A Check Digit
     * default = enable
     * range = 0:disable, 1:enable
     */
    public static final int TRANSMIT_UPC_A_CHK_DIGIT = 0x0028;

    /**
     * Transmit UPC-E Check Digit
     * default = enable
     * range = 0:disable, 1:enable
     */
    public static final int TRANSMIT_UPC_E_CHK_DIGIT = 0x0029;

    /**
     * Transmit UPC-E1 Check Digit
     * default = enable
     * range = 0:disable, 1:enable
     */
    public static final int TRANSMIT_UPC_E1_CHK_DIGIT = 0x002A;

    /**
     * UPC-A Preamble
     * default = System Character
     * range = 0:No Preamble, 1:System Character, 2: System Character& Country Code
     * Preamble characters are part of the UPC symbol, and include Country Code and System Character.
     * There are three options for transmitting a UPC-A preamble to the host device: transmit System Character
     * only,
     * transmit System Character and Country Code ("0" for USA), and transmit no preamble.
     * Select the option that matches the host system.
     */
    public static final int UPC_A_PREAMBLE = 0x0022;

    /**
     * UPC-E Preamble
     * default = System Character
     * range = 0:No Preamble, 1:System Character, 2: System Character& Country Code
     * Preamble characters are part of the UPC symbol, and include Country Code and System Character.
     * There are three options for transmitting a UPC-E preamble to the host device: transmit System Character
     * only,
     *   transmit System Character and Country Code ("0" for USA), and transmit no preamble.
     *   Select the option that matches the host system.
     */
    public static final int UPC_E_PREAMBLE = 0x0023;

    /**
     * UPC-E1 Preamble
     * default = System Character
     * range = 0:No Preamble, 1:System Character, 2: System Character& Country Code
     * Preamble characters are part of the UPC symbol, and include Country Code and System Character.
     * There are three options for transmitting a UPC-E1 preamble to the host device: transmit System Character
     * only,
     * transmit System Character and Country Code ("0" for USA), and transmit no preamble.
     * Select the option that matches the host system.
     */
    public static final int UPC_E1_PREAMBLE = 0x0024;
```

```
/**
 * Convert UPC-E to UPC-A
 * default = disable
 * range = 0:disable, 1:enable
 */
public static final int CONVERT_UPC_E_TO_A = 0x0025

/**
 * Convert UPC-E1 to UPC-A
 * default = disable
 * range = 0:disable, 1:enable
 */
public static final int CONVERT_UPC_E1_TO_A = 0x0026

/**
 * EAN-8/JAN-8 Extend
 * default = disable
 * range = 0:disable, 1:enable
 */
public static final int EAN8_EXTEND = 0x0027

/**
 * Coupon Report
 * default = New Coupon Symbols
 * range = 0:Old Coupon Symbol, 1:New Coupon Symbols, 2:Both Coupon Formats
 * Old Coupon Symbols - Scanning an old coupon symbol reports both UPC and Code 128, scanning an
 * interim coupon symbol reports UPC, and scanning a new coupon symbol reports nothing (no decode).
 * New Coupon Symbols - Scanning an old coupon symbol reports either UPC or Code 128, and scanning
 * an interim coupon symbol or a new coupon symbol reports Databar Expanded.
 * Both Coupon Formats - Scanning an old coupon symbol reports both UPC and Code 128, and scanning
 * an interim coupon symbol or a new coupon symbol reports Databar Expanded.
 */
public static final int COUPON_REPORT = 0xF1DA

/**
 * ISSN EAN
 * default = disable
 * range = 0:disable, 1:enable
 */
public static final int ISSN_EAN = 0xF169
```

```
/**
 * Code 128
 * default = enable
 * range = 0:disable, 1:enable
 */
public static final int CODE128 = 0x0008

/**
 * Set Lengths for Code 128
 * default = 3
 * range = 0 ~ 255
 */
public static final int CODE128_LEN_MIN = 0x00D1;

/**
 * Set Lengths for Code 128
 * default = 30
 * range = 0 ~ 255
 */
public static final int CODE128_LEN_MAX = 0x00D2;

/**
 * GS1-128 (formerly UCC/EAN-128)
 * default = enable
 * range = 0:disable, 1:enable
 */
public static final int GS1_128 = 0x000E;

/**
 * ISBT 128
 * default = enable
 * range = 0:disable, 1:enable
 */
public static final int ISBT128 = 0x0054

/**
 * ISBT Concatenation
 * default = disable
 * range = 0:Disable ISBT Concatenation, 1:Enable ISBT Concatenation, 2:Autodiscriminate ISBT
 * Concatenation
 */
```

```
    public static final int ISBT128_CONCATENATION = 0xF141;

    /**
     * Check ISBT Table
     * default = enable
     * range = 0:disable, 1:enable
     */
    public static final int ISBT128_CHECK_TABLE = 0xF142;

    /**
     * ISBT Concatenation Redundancy
     * default = 10
     * range = 2 ~ 20
     * If you set ISBT Concatenation to Autodiscriminate,
     * use this parameter to set the number of times the decoder must decode an ISBT symbol before
     * determining that there is no additional symbol.
     */
    public static final int ISBT128_CONCATENATION_REDUNDANCY = 0x00DF;

    /**
     * Code 39
     * default = enable
     * range = 0:disable, 1:enable
     */
    public static final int CODE39 = 0x0000;

    /**
     * Trioptic Code 39
     * default = enable
     * range = 0:disable, 1:enable
     */
    public static final int TRIOPTIC_CODE39 = 0x000D;

    /**
     * Convert Code 39 to Code 32
     * default = disable
     * range = 0:disable, 1:enable
     */
    public static final int CONVERT_CODE39_32 = 0x0056;

    /**
```

```
 * Code 32 Prefix
 * default = enable
 * range = 0:disable, 1:enable
 */
public static final int CODE32_PREFIX = 0x00E7;

/**
 * Set Lengths for Code 39
 * default = 3
 * range = length within range: 0 ~ 255
 */
public static final int CODE39_LEN_MIN = 0x0012;

/**
 * Set Lengths for Code 39
 * default = 30
 * range = length within range: 0 ~ 255
 */
public static final int CODE39_LEN_MAX = 0x0013;

/**
 * Code 39 Check Digit Verification
 * default = disable
 * range = 0:disable, 1:enable
 */
public static final int CODE39_CHK_DIGIT_VERIFICATION = 0x0030;

/**
 * Transmit Code 39 Check Digit
 * default = disable
 * range = 0:disable, 1:enable
 */
public static final int TRANSMIT_CODE39_CHK_DIGIT = 0x002B;

/**
 * Code 39 Full ASCII Conversion
 * default = disable
 * range = 0:disable, 1:enable
 */
public static final int CODE39_FULL_ASCII = 0x0011;
```

```
        /**
         * Code 93
         * default = enable
         * range = 0:disable, 1:enable
         */
        public static final int CODE93 = 0x0009;

        /**
         * Set Lengths for Code 93
         * default = 3
         * range = Length Within Range: 0 ~ 255
         */
        public static final int CODE93_LEN_MIN = 0x001A;

        /**
         * Set Lengths for Code 93
         * default = 30
         * range = Length Within Range: 0 ~ 255
         */
        public static final int CODE93_LEN_MAX = 0x001B;

        /**
         * Code 11
         * default = enable
         * range = 0:disable, 1:enable
         */
        public static final int CODE11 = 0x000A;

        /**
         * Set Lengths for Code 11
         * default = 3
         * range = Length Within Range: 0 ~ 255
         */
        public static final int CODE11_LEN_MIN = 0x001C;

        /**
         * Set Lengths for Code 11
         * default = 30
         * range = Length Within Range: 0 ~ 255
         */
        public static final int CODE11_LEN_MAX = 0x001D;
```

```
/**
 * Code 11 Check Digit Verification
 * default = disable
 * range = 0:Disable, 1:One Check Digit, 2:Two Check Digits
 */
public static final int CODE11_CHK_DIGIT_VERIFICATION = 0x0034;

/**
 * Transmit Code 11 Check Digits
 * default = disable
 * range = 1:Transmit Code 11 Check Digit(s) (Enable), 0:Do Not Transmit Code 11 Check Digit(s) (Disable)
 */
public static final int TRANSMIT_CODE11_CHK_DIGIT = 0x002F;

/**
 * Interleaved 2 of 5 (ITF)
 * default = disable
 * range = 0:disable, 1:enable
 */
public static final int INTERLEAVED2OF5 = 0x0006;

/**
 * Set Lengths for Interleaved 2 of 5
 * default = 3
 * range = Length Within Range: 0 ~ 55
 */
public static final int INTERLEAVED2OF5_LEN_MIN = 0x0016;

/**
 * Set Lengths for Interleaved 2 of 5
 * default = 30
 * range = Length Within Range: 0 ~ 55
 */
public static final int INTERLEAVED2OF5_LEN_MAX = 0x0017;

/**
 * I 2 of 5 Check Digit Verification
 * default = disable
 * range = 0:disable, 1:USS Check Digit, 2:OPCC Check Digit
 */
```

```
    public static final int INTERLEAVED2OF5_CHK_DIGIT = 0x0031;

    /**
     * Transmit I 2 of 5 Check Digit
     * default = disable
     * range = 0:disable, 1:enable
     */
    public static final int TRANSMIT_INTERLEAVED2OF5_CHK_DIGIT = 0x002C;

    /**
     * Convert I 2 of 5 to EAN-13
     * default = disable
     * range = 0:disable, 1:enable
     */
    public static final int CONVERT_INTERLEAVED_EAN13 = 0x0052;

    /**
     * Discrete 2 of 5 (DTF)
     * default = enable
     * range = 0:disable, 1:enable
     */
    public static final int DISCRETE2OF5 = 0x0005;

    /**
     * Set Lengths for Discrete 2 of 5
     * default = 3
     * range = Length Within Range: 0 ~ 55
     */
    public static final int DISCRETE2OF5_LEN_MIN = 0x0014;

    /**
     * Set Lengths for Discrete 2 of 5
     * default = 30
     * range = Length Within Range: 0 ~ 55
     */
    public static final int DISCRETE2OF5_LEN_MAX = 0x0015;

    /**
     * Codabar (NW - 7)
     * default = enable
     * range = 0:disable, 1:enable
```

```
 */
public static final int CODABAR = 0x0007;

/**
 * Set Lengths for Codabar
 * default = 3
 * range = Length Within Range: 0 ~ 255
 */
public static final int CODABAR_LEN_MIN = 0x0018;

/**
 * Set Lengths for Codabar
 * default = 30
 * range = Length Within Range: 0 ~ 255
 */
public static final int CODABAR_LEN_MAX = 0x0019;

/**
 * CLSI Editing
 * default = disable
 * range = 0:disable, 1:enable
 */
public static final int CODABAR_CLSI_EDIT = 0x0036;

/**
 * NOTIS Editing
 * default = disable
 * range = 0:disable, 1:enable
 */
public static final int CODABAR_NOTIS_EDIT = 0x0037;

/**
 * MSI
 * default = enable
 * range = 0:disable, 1:enable
 */
public static final int MSI = 0x000B;

/**
 * Set Lengths for MSI
 * default = 3
```

```
     * range = Length Within Range: 0 ~ 255
     */
    public static final int MSI_LEN_MIN = 0x001E;

    /**
     * Set Lengths for MSI
     * default = 30
     * range = Length Within Range: 0 ~ 255
     */
    public static final int MSI_LEN_MAX = 0x001F;

    /**
     * MSI Check Digits
     * default = One MSI Check Digit
     * range = 0:One MSI Check Digit, 1:Two MSI Check Digits
     */
    public static final int MSI_CHK_DIGIT = 0x0032;

    /**
     * Transmit MSI Check Digit(s)
     * default = disable
     * range = 0:Do Not Transmit MSI Check Digit(s) (Disable), 1:Transmit MSI Check Digit(s) (Enable)
     */
    public static final int TRANSMIT_MSI_CHK_DIGIT = 0x002E;

     /**
     * MSI Check Digit Algorithm
     * default = MOD 10/10
     * range = 0:MOD 10/MOD 11, 1:MOD 10/MOD 10
     */
    public static final int MSI_CHK_DIGIT_ALGORITHM = 0x0033;

    /**
     * Chinese 2 of 5
     * default = enable
     * range = 0:disable, 1:enable
     */
    public static final int CHINESE_POST = 0xF098;

    /**
     * Matrix 2 of 5
     * default = enable
     * range = 0:disable, 1:enable
     */
    public static final int MATRIX2OF5 = 0xF16A;

    /**
     * Set Lengths for Matrix 2 of 5
     * default = 3
     * range = Length Within Range: 0 ~ 255
     */
    public static final int MATRIX2OF5_LEN_MIN = 0xF16B;

    /**
     * Set Lengths for Matrix 2 of 5
     * default = 30
     * range = Length Within Range: 0 ~ 255
     */
    public static final int MATRIX2OF5_LEN_MAX = 0xF16C;

    /**
     * Matrix 2 of 5 Check Digit
     * default = disable
     * range = 0:disable, 1:enable
     */
    public static final int MATRIX2OF5_CHK_DIGIT = 0xF16E;

     /**
     * Transmit Matrix 2 of 5 Check Digit
     * default = disable
     * range = 0:disable, 1:enable
     */
    public static final int TRANSMIT_MATRIX2OF5_CHK_DIGIT = 0xF16F;

    /**
     * Korean 3 of 5
     * default = enable
     * range = 0:disable, 1:enable
     */
    public static final int KOREA_POST = 0xF145;

    /**
```

```
 * Inverse 1D
 * default = Regular
 * range = 0:Regular, 1:Inverse Only, 2:Inverse Autodetect
 */
public static final int INVERSE_1D = 0xF14A;

/**
 * US Postnet
 * default = enable
 * range = 0:disable, 1:enable
 */
public static final int US_POSTNET = 0x0059;

/**
 * US Planet
 * default = enable
 * range = 0:disable, 1:enable
 */
public static final int US_PLANET = 0x005A;

/**
 * Transmit US Postal Check Digit
 * default = enable
 * range = 0:disable, 1:enable
 */
public static final int TRANSMIT_US_POSTNET_CHK_DIGIT = 0x005F;

/**
 * UK Postal
 * default = enable
 * range = 0:disable, 1:enable
 */
public static final int UK_POST = 0x005B;

/**
 * Transmit UK Postal Check Digit
 * default = enable
 * range = 0:disable, 1:enable
 */
public static final int TRANSMIT_UK_POST_CHK_DIGIT = 0x0060;
```

```
        /**
         * Japan Postal
         * default = enable
         * range = 0:disable, 1:enable
         */
        public static final int JAPANESE_POST = 0xF022;

        /**
         * Australia Post
         * default = enable
         * range = 0:disable, 1:enable
         */
        public static final int AUSTRAILA_POST = 0xF023;

        /**
         * Australia Post Format
         * default = Autodiscriminate
         * range = 0:Autodiscriminate, 1:Raw Format, 2:Alphanumeric Encoding, 3:Numeric Encoding
         */
        public static final int AUSTRAILA_POST_FORMAT = 0xF1CE;

        /**
         * Netherlands KIX Code
         * default = enable
         * range = 0:disable, 1:enable
         */
        public static final int NETHELANS_POST = 0xF046;

       /**
         * USPS 4CB/One Code/Intelligent Mail
         * default = enable
         * range = 0:disable, 1:enable
         */
        public static final int USPS_4 = 0xF150;

        /**
         * UPU FICS Postal
         * default = enable
         * range = 0:disable, 1:enable
         */
        public static final int UPI_FICS_POST = 0xF163;
```

```
    /**
     * GS1 DataBar
     * default = enable
     * range = 0:disable, 1:enable
     */
    public static final int GS1_DATABAR = 0xF052;

    /**
     * GS1 DataBar Limited
     * default = disable
     * range = 0:disable, 1:enable
     */
    public static final int GS1_LIMIT = 0xF053;

    /**
     * GS1 DataBar Limited Security Level
     * default = 3
     * range = 1:Security Level 1, 2:Security Level 2, 3:Security Level 3, 4:Security Level 4
     * Level 1 - No clear margin required.
     * This complies with the original GS1 standard,
     * yet might result in erroneous1 decoding of the DataBar Limited bar code when scanning some UPC
     * symbols that start with the digits "9" and "7".
     * Level 2 - Automatic risk detection.
     * This level of security may result in erroneous decoding of DataBar Limited bar codes when scanning
     * some UPC symbols.
     * If a misdecode is detected, the decoder operates in Level 3 or Level 1.
     * Level 3 - Security level reflects newly proposed GS1 standard that requires a 5X trailing clear margin.
     * Level 4 - Security level extends beyond the standard required by GS1.
     * This level of security requires a 5X leading and trailing clear margin.
     */
    public static final int GS1_LIMIT_SECURITY = 0xF1D8;

    /**
     * GS1 DataBar Expanded
     * default = disable
     * range = 0:disable, 1:enable
     */
    public static final int GS1_EXPAND = 0xF054;

    /**
```

```
     * Convert GS1 DataBar to UPC/EAN
     * default = disable
     * range = 0:disable, 1:enable
     */
    public static final int CONVERT_GS1_UPCEAN = 0xF08D;

    /**
     * Composite CC-C
     * default = disable
     * range = 0:disable, 1:enable
     */
    public static final int COMPOSIT_CC_C = 0xF055;

    /**
     * Composite CC-A/B
     * default = disable
     * range = 0:disable, 1:enable
     */
    public static final int COMPOSIT_CC_AB = 0xF056;

    /**
     * Composite TLC-39
     * default = disable
     * range = 0:disable, 1:enable
     */
    public static final int COMPOSIT_TCL39 = 0xF073;

    /**
     * UPC Composite Mode
     * default = UPC Always Linked
     * range = 0:UPC Never Linked, 1:UPC Always Linked, 2:Autodiscriminate UPC Composites
     */
    public static final int UPC_COMPOSIT_MODE = 0xF058;

    /**
     * GS1-128 Emulation Mode for UCC/EAN Composite Codes
     * default = disable
     * range = 0:disable, 1:enable
     */
    public static final int GS1128_EMULATION_MODE = 0xF0AB;
```

```
/**
 * PDF417
 * default = enable
 * range = 0:disable, 1:enable
 */
public static final int PDF417 = 0x000F;

/**
 * MicroPDF417
 * default = enable
 * range = 0:disable, 1:enable
 */
public static final int MICRO_PDF = 0x00E3;

/**
 * Data Matrix
 * default = enable
 * range = 0:disable, 1:enable
 */
public static final int DATA_MATRIX = 0xF024;

/**
 * Data Matrix Inverse
 * default = Regular
 * range = 0:Regular, 1:Inverse Only, 2: Inverse Autodetect
 */
public static final int DATA_MATRIX_INVERSE = 0xF14C;

/**
 * Decode Mirror Images (Data Matrix Only)
 * default = Never
 * range = 0:Never, 1:Always, 2:Auto
 */
public static final int DATA_MATRIX_DECODE_MIRROR = 0xF119;

/**
 * Maxicode
 * default = enable
 * range = 0:disable, 1:enable
 */
public static final int MAXICODE = 0xF026;
```

```
/**
 * QR Code
 * default = enable
 * range = 0:disable, 1:enable
 */
public static final int QR_CODE = 0xF025;

/**
 * MicroQR
 * default = enable
 * range = 0:disable, 1:enable
 */
public static final int MICRO_QR = 0xF13D;

/**
 * Aztec
 * default = enable
 * range = 0:disable, 1:enable
 */
public static final int AZTECCODE = 0xF13E;

/**
 * Aztec Inverse
 * default = Regular
 * range = 0:Regular, 1:Inverse Only, 2:Inverse Autodetect
 */
public static final int AZTECCODE_INVERSE = 0xF14D;

/**
 * Han Xin
 * default = enable
 * range = 0:disable, 1:enable
 */
public static final int HANXIN = 0xF8048F;

/**
 * Han Xin Inverse
 * default = Regular
 * range = 0:Regular, 1:Inverse Only, 2:Inverse Autodetect
 */
```

```
public static final int HANXIN_INVERSE = 0xF80490;

/**
 * Redundancy Level
 * default = Redundancy 1
 * range = 1:Redundancy level 1, 2:Redundancy level 2, 3:Redundancy level 3, 4:Redundancy level 4
 * Level1 : Codebar(code type) =   8 characters or less(length)
 *              MSI(code type) =   4 characters or less(length)
 *              D 2 of 5(code type) =   8 characters or less(length)
 *              I 2 of 5(code type) =   8 characters or less(length)
 * Level2 :   All(code type) = All(length)
 * Level3 : MSI(code type) =   4 characters or less(length)
 *              D 2 of 5(code type) =   8 characters or less(length)
 *              I 2 of 5(code type) =   8 characters or less(length)
 *              Codebar(code type) =   8 characters or less(length)
 * Level4 :   All(code type) = All(length)
 */
public static final int REDUNDANCY_LEVEL = 0x004E;

/**
 * Security Level
 * default = Security level 1
 * range = 0:Security level 0, 1:Security level 1, 2:Security level 2, 3:Security level 3
 * Security Level 0: This setting allows the decoder to operate in its most aggressive state,
 * while providing sufficient security in decoding most "in-spec" bar codes.
 * Security Level 1: This default setting eliminates most misdecodes.
 * Security Level 2: Select this option if Security level 1 fails to eliminate misdecodes.
 * Security Level 3: If you selected Security Level 2 and misdecodes still occur,
 * select this security level. Be advised, selecting this option is an extreme measure against mis-decoding
 * severely out of spec bar codes.
 * Selecting this level of security significantly impairs the decoding ability of the decoder.
 * If you need this level of security, try to improve the quality of the bar codes.
 */
public static final int SECURITY_LEVEL = 0x004D;

/**
 * Intercharacter Gap Size
 * default = Normal Intercharacter Gaps
 * range = 06:Normal Intercharacter Gaps , 0A:Large Intercharacter Gaps
 */
public static final int INTERCHARATER_GAP_SIZE = 0xF07D;
```

```
        /**
         * Decode Session Timeout
         * default = 30 (3 sec)
         * range = 5 ~ 99 (ex 5sec = 50(integer). 5.3sec = 53(integer)
         */
        public static final int DECODE_TIMEOUT = 0x0088;

        /**
         * Timeout Between Decodes, Same Symbol
         * default = 6 (0.6 sec)
         * range = 0 ~ 99 (ex 5sec = 50(integer). 5.3sec = 53(integer)
         */
        public static final int TIMEOUT_SAME_SYMBOL = 0x0089;

        /**
         * Decode Aiming Pattern
         * default = enable
         * range = 0:disable, 2:enable
         */
        public static final int AIMER_MODE = 0xF032;

        /**
         * Decoding Illumination
         * default = enable
         * range = 0:disable, 1:enable
         */
        public static final int ILLUMINATION_MODE = 0xF02A;

        /**
         * Picklist Mode
         * default = disable
         * range = 0:disable, 2:enable
         */
        public static final int PICKLIST_MODE = 0xF092;
    }
```

- **BB Barcode Symbology values**

: SymbologyType class

```
    public static class SymbologyType {
        public static final int SYMBOLOGY_UNKNOWN = -1;
        public static final int SYMBOLOGY_UPC_A = 1;

        public static final int SYMBOLOGY_UPC_E = 2;

        public static final int SYMBOLOGY_UPC_E1 = 3;

        public static final int SYMBOLOGY_EAN8 = 4;

        public static final int SYMBOLOGY_EAN13 = 5;

        public static final int SYMBOLOGY_BOOKLAND = 6;

        public static final int SYMBOLOGY_CODE39 = 8;

        public static final int SYMBOLOGY_CODE93 = 9;

        public static final int SYMBOLOGY_CODE128 = 10;

        public static final int SYMBOLOGY_INTERLEAVED2OF5 = 11;

        public static final int SYMBOLOGY_CODABAR = 12;

        public static final int SYMBOLOGY_CODE11 = 13;

        public static final int SYMBOLOGY_MSI = 14;

        public static final int SYMBOLOGY_PDF417 = 16;

        public static final int SYMBOLOGY_ISBT128 = 17;

        public static final int SYMBOLOGY_MATRIX2OF5 = 19;

        public static final int SYMBOLOGY_DATAMATRIX = 20;

        public static final int SYMBOLOGY_MAXICODE = 21;

        public static final int SYMBOLOGY_AZTECCODE = 22;

        public static final int SYMBOLOGY_MICROPDF = 23;
        public static final int SYMBOLOGY_QRCODE = 24;

        public static final int SYMBOLOGY_TRIOPTIC_CODE = 25;

        public static final int SYMBOLOGY_DISCRETE2OF5 = 26;

        public static final int SYMBOLOGY_USPS4CB = 27;

        public static final int SYMBOLOGY_AUSTRALIA_POST = 28;

        public static final int SYMBOLOGY_UK_POST = 29;

        public static final int SYMBOLOGY_CHINESE_POST = 30;

        public static final int SYMBOLOGY_JAPANESE_POST = 31;

        public static final int SYMBOLOGY_NETHERLANDS_POST = 32;

        public static final int SYMBOLOGY_KOREAN_POST = 33;

        public static final int SYMBOLOGY_US_POSTNET = 34;

        public static final int SYMBOLOGY_US_PLANET = 35;

        public static final int SYMBOLOGY_EAN_TRANSMIT_ISSN = 43;

        public static final int SYMBOLOGY_CODE39_FULL_ASCII = 51;

        public static final int SYMBOLOGY_GS1_LIMITED = 74;

        public static final int SYMBOLOGY_ISBT128_CONCATENATION = 76;

        public static final int SYMBOLOGY_COMPOSITE_TLC_39 = 85;

        public static final int SYMBOLOGY_COUPON_REPORT = 95;

        public static final int SYMBOLOGY_GS1_DATABAR_EXPANDED = 101;

        public static final int SYMBOLOGY_UPU_FICS_POSTAL = 104;

        public static final int SYMBOLOGY_MICROQR = 113;
```

```
public static final int SYMBOLOGY_CODE49 = 114;

public static final int SYMBOLOGY_OCR = 115;

public static final int SYMBOLOGY_CANADIAN_POST = 116;

public static final int SYMBOLOGY_CODE32 = 119;

public static final int SYMBOLOGY_CODE16K = 123;

public static final int SYMBOLOGY_HANXIN = 134;

public static final int SYMBOLOGY_IATA = 136;

public static final int SYMBOLOGY_EAN128 = 137;

public static final int SYMBOLOGY_UPC_D = 138;

public static final int SYMBOLOGY_GS1_DATABAR = 139;

public static final int SYMBOLOGY_SCANLET = 140;

public static final int SYMBOLOGY_CUECODE = 141;

public static final int SYMBOLOGY_SIGNATURE_CAPTURE = 142;

public static final int SYMBOLOGY_FRENCH_LOT = 149;

public static final int SYMBOLOGY_PARAMETER_FNC3 = 151;

public static final int SYMBOLOGY_MULTI_PKT_FORM = 179;

public static final int SYMBOLOGY_GS1_DATAMATIRIX = 182;

public static final int SYMBOLOGY_GS1_QR = 183;

public static final int SYMBOLOGY_RFID_RAW = 184;

public static final int SYMBOLOGY_RFID_URI = 185;
```

```
    }
```

BB Sled BT callback command message values

: BTCmdMsg class

※ Only for Bluetooth interface (BTReader)

```
    public static class BTCmdMsg {
        /**
         * This callback message occur when SLED found device after scanning and always contains bundle data.
         * 1) Name - Key : ConstantsBT.BT_BUNDLE_NAME_KEY, Format : String
         * 2) Address - Key : ConstantsBT.BT_BUNDLE_ADDR_KEY, Format : String
         * 3) Bond state - Key : ConstantsBT.BT_BUNDLE_BOND_STATE_KEY, Format : Int
         */
        public static final int SLED_BT_DEVICE_FOUND = 52;

        /**
         * This callback message occur when Pairing status change and always contains bundle data.
         * 1) Name - Key : ConstantsBT.BT_BUNDLE_NAME_KEY, Format : String
         * 2) Address - Key : ConstantsBT.BT_BUNDLE_ADDR_KEY, Format : String
         * 3) Bond state - Key : ConstantsBT.BT_BUNDLE_BOND_STATE_KEY, Format : Int
         * 4) Bond new state - Key : ConstantsBT.BT_BUNDLE_BOND_NEW_STATE_KEY, Format : Int
         * 5) Bond previous state - Key : ConstantsBT.BT_BUNDLE_BOND_PREV_STATE_KEY, Format : Int
         */
        public static final int SLED_BT_BOND_STATE_CHAGNED = 53;

        /**
         * This callback message occur when Receive pair request and not contains bundle data.
         */
        public static final int SLED_BT_PAIRING_REQUEST = 54;

        /**
         * This callback message occur when Start the scan and not contains bundle data.
         */
        public static final int SLED_BT_DISCOVERY_STARTED = 55;

        /**
         * This callback message occur when The scan complete and not contains bundle data
         */
        public static final int SLED_BT_DISCOVERY_FINISHED = 56;

        /**
```

```
     * This callback message occur when Changed device's Bluetooth state and always contains bundle data.
     * 1) Bond new state - Key : ConstantsBT.BT_BUNDLE_BOND_NEW_STATE_KEY, Format : Int
     * 2) Bond previous state - Key : ConstantsBT.BT_BUNDLE_BOND_PREV_STATE_KEY, Format : Int
     */
    public static final int SLED_BT_STATE_CHANGED = 57;

    public static final int SLED_BT_CONNECTION_STATE_CHANGED = 58;

    public static final int SLED_BT_ADAPTER_CONNECTION_STATE_CHANGED = 59;

    public static final int SLED_BT_CONNECTION_ESTABLISHED = 60;

    public static final int SLED_BT_DISCONNECTED = 61;

    public static final int SLED_BT_CONNECTION_LOST = 62;

    /**
     * This callback message occur when Changing to a connected state and always contains bundle data.
     * 1) Name - Key : ConstantsBT.BT_BUNDLE_NAME_KEY, Format : String
     * 2) Address - Key : ConstantsBT.BT_BUNDLE_ADDR_KEY, Format : String
     * 3) Bond state - Key : ConstantsBT.BT_BUNDLE_BOND_STATE_KEY, Format : Int
     */
    public static final int SLED_BT_ACL_CONNECTED = 63;

    /**
     * This callback message occur when Receive disconnect request and always contains bundle data.
     * 1) Name - Key : ConstantsBT.BT_BUNDLE_NAME_KEY, Format : String
     * 2) Address - Key : ConstantsBT.BT_BUNDLE_ADDR_KEY, Format : String
     * 3) Bond state - Key : ConstantsBT.BT_BUNDLE_BOND_STATE_KEY, Format : Int
     */
    public static final int SLED_BT_ACL_DISCONNECT_REQUESTED = 64;

    /**
     * This callback message occur when Changing to a disconnect state and always contains bundle data.
     * Bundle data
     * 1) Name - Key : ConstantsBT.BT_BUNDLE_NAME_KEY, Format : String
     * 2) Address - Key : ConstantsBT.BT_BUNDLE_ADDR_KEY, Format : String
     * 3) Bond state - Key : ConstantsBT.BT_BUNDLE_BOND_STATE_KEY, Format : Int
     */
    public static final int SLED_BT_ACL_DISCONNECTED = 65;
}
```

■ BB Sled Get command common result values

: BTCommonResult class

※ Only for Bluetooth interface (BTReader)

```
        public static final int BT_NOT_ENABLE_STATE   = -40;

        public static final int ACCESS_TIMEOUT = -32;

        public static final int ALREADY_CONNECTING = -18;

        public static final int COMMUNICATION_ERROR = -16;

        public static final int BLUETOOTH_NOT_ENABLED = -15;

        public static final int CHARGING_STATE_ERROR = -14;

        public static final int FILE_IS_NOT_EXIST = -13;

        public static final int LOW_BATTERY = -12;

        public static final int ALREADY_CONNECTED = -10;

        public static final int ALREADY_DISCONNECTED = -9;

        public static final int DUP_CMD_ERROR = -8;

        public static final int READER_OR_SERIAL_STATUS_ERROR = -7;

        public static final int MODE_ERROR = -6;

        public static final int SD_NOT_CONNECTED = -5;

        public static final int OTHER_CMD_RUNNING_ERROR = -4;

        public static final int ARGUMENT_ERROR = -3;

        public static final int OTHER_ERROR = -1;
```

- **BB Sled Set command result values**

: BTResult class

※ Only for Bluetooth interface (BTReader)

```
    public static class BTResult extends BTCommonResult {
        ~~public static final int BT_NOT_ENABLE_STATE   =  -40;~~


        public static final int SUCCESS   = 0;
    }
```

BB Sled BT State

: BTState class

※ Only for Bluetooth interface (BTReader)

```
    public static class BTState {
        public static final int STATE_OFF = 10;


        public static final int STATE_TURNING_ON = 11;


        public static final int STATE_ON = 12;


        public static final int STATE_TURNING_OFF = 13;
    }
```

BB Sled BT Bond State

: BTBondState class

※ Only for Bluetooth interface (BTReader)

```
    public static class BTBondState {
        public static final int BOND_NONE = 10;


        public static final int BOND_BONDING = 11;


        public static final int BOND_BONDED = 12;
    }
```

BB Sled Set command result values

: BTConnectState class

※ Only for Bluetooth interface (BTReader)

```
    public static class BTConnectState extends BTCommonResult {
        public static final int NONE = 0;


        public static final int CONNECTING = 1;


        public static final int CONNECTED = 2;
    }
```

■ BB Sled Bluetooth device type

: BTDeviceType class

※ Only for Bluetooth interface (BTReader)

```
public static class BTDeviceType {
    public static final String TYPE_1 = “01”; //RFR900 Classic

    public static final String TYPE_2 = “02”; //RFR901 Classic

    public static final String TYPE_3 = “03”; //RFR901 LE
}
```

3) Default

■ Default Value(ResetConfigToFactoryDefaults API)

[Access timeout] = 3000

[Access password] = 0

[Duty]= 100

[Turbo mode] = 1(On)

[Power] = 30(dBM)

[Singulation] = 4

[RSSI] = 1(On)

[Session] = 0

[Toggle] = 1(On)

[Inventory session target] = 0(Target A)

4) Arguments

■ Describe arguments of RF APIs

int RFMemType;

The memory bank type. 0=RESERVED, 1=EPC, 2=TID, 3=USER

int startlocation;

The first starting point(word base). 1word is 16bits.

int length;

The number of bits in the mask. Valid values are 0 to 255.

String mask;(Maximum 64 length)

A buffer that contains a left-justified bit array that represents that bit pattern to Match

- HEX format of Selection Mask. Nibble Unit(4-bits)
- Set separately bank, offset(bit unit), action
- In case of non-target, enter NULL(null)

5) Event Handler

Events	Value	Descriptions
SDConsts.RFCmdMsg.RESPONSE_CODE	20	Reserved
SDConsts.RFCmdMsg.UNKNOWN	50	Reserved
SDConsts.SDCmdMsg.SLED_BATTERY_STATE_CHANGE	43	Sled update message
SDConsts.SDCmdMsg.SLED_MODE_CHANGED	45	Sled update message
SDConsts.SDCmdMsg.INVENTORY_STATE_CHANGED	46	Sled update message

6) Barcode mode

- ● Can change to barcode mode with sled hardware's mode button.
  - Mode button is located on the left.
- ● Only use after connect the sled with SD_Connect API.
- ● After mode change, can get the each trigger mode
  - (0 : RFID / 1 : BARCODE)
- ● Reference some constants related with barcode mode
  - BCCmdMsg class(※Reference 3.2.G)

7) Barcode parameters

No.	Param Name	SB Param value	Default setting value	Setting value range
1	UPC-A	0x0001	1 : enable	0 : disable 1 : enable
2	UPC-E	0x0002	1 : enable	0 : disable 1 : enable
3	UPC-E1	0x000C	1 : enable	0 : disable 1 : enable
4	EAN-8/JAN-8	0x0004	1 : enable	0 : disable 1 : enable
5	EAN-13/JAN-13	0x0003	1 : enable	0 : disable 1 : enable
6	Bookland EAN	0x0053	1 : enable	0 : disable 1 : enable
7	Bookland ISBN Format	0xF140	1 : enable	0 : Bookland ISBN-10 1 : Bookland ISBN-13
8	Decode UPC/EAN/JAN Supplementals	0x0010	0 : Ignore Supplemental	0 : Ignore Supplemental 1 : Decode UPC/EAN/JAN Only With Supplementals 2 : Autodiscriminate UPC/EAN/JAN Supplementals 3 : Enable Smart Supplemental Mode 4 : Enable 378/379 Supplemental Mode 5 : Enable 978/979 Supplemental Mode 6 : Enable 414/419/434/439 Supplemental Mode 7 : Enable 977 Supplemental Mode 8 : Enable 491 Supplemental Mode 9 : Supplemental User-Programmable Type 1 A : Supplemental User-Programmable Type 1 and 2 B : Smart Supplemental Plus User-Programmable 1 C : Smart Supplemental Plus User-Programmable 1 and 2
9	UPC/EAN/JAN Supplemental Redundancy	0x0050	10	2 ~ 16
10	UPC/EAN/JAN Supplemental AIM ID Format	0xF1A0	1 : combined	0 : separate 1 : combined 2 : separate transmissions
11	Transmit UPC-A Check Digit	0x0028	1 : enable	0 : disable 1 : enable
12	Transmit UPC-E Check Digit	0x0029	1 : enable	0 : disable 1 : enable
13	Transmit UPC-E1 Check Digit	0x002A	1 : enable	0 : disable 1 : enable
14	UPC-A Preamble	0x0022	1 : System Character	0 : No Preamble 1 : System Character 2 : System Character & Country Code
15	UPC-E Preamble	0x0023	1 : System Character	0 : No Preamble 1 : System Character 2 : System Character & Country Code
16	UPC-E1 Preamble	0x0024	1 : System Character	0 : No Preamble 1 : System Character 2 : System Character & Country Code
17	Convert UPC-E to UPC-A	0x0025	0 : disable	0 : disable 1 : enable
18	Convert UPC-E1 to UPC-A	0x0026	0 : disable	0 : disable 1 : enable
19	EAN-8/JAN-8 Extend	0x0027	0 : disable	0 : disable 1 : enable
20	Coupon Report	0xF1DA	1 : New Coupon Symbol	0 : Old Coupon Symbol 1 : New Coupon Symbols 2 : Both Coupon Formats
21	ISSN EAN	0xF169	0 : disable	0 : disable 1 : enable
22	Code 128	0x0008	1 : enable	0 : disable 1 : enable
23	Set Lengths for Code 128	0x00D1	3	0 ~ 255
24		0x00D2	30	
25	GS1-128 (formerly UCC/EAN-128)	0x000E	1 : enable	0 : disable 1 : enable

No.	Param Name	SB Param value	Default setting value	Setting value range
26	ISBT 128	0x0054	1 : enable	0 : disable 1 : enable
27	ISBT Concatenation	0xF141	0 : Disable ISBT Concate	0 : Disable ISBT Concatenation 1 : Enable ISBT Concatenation 2 : Autodiscriminate ISBT Concatenation
28	Check ISBT Table	0xF142	1 : enable	0 : disable 1 : enable
29	ISBT Concatenation Redundancy	0x00DF	10	2 ~ 20
30	Code 39	0x0000	1 : enable	0 : disable 1 : enable
31	Trioptic Code 39	0x000D	1 : enable	0 : disable 1 : enable
32	Convert Code 39 to Code 32	0x0056	0 : disable	0 : disable 1 : enable
33	Code 32 Prefix	0x00E7	1 : enable	0 : disable 1 : enable
34	Set Lengths for Code 39	0x0012	3	0 ~ 255
35		0x0013	30	
36	Code 39 Check Digit Verification	0x0030	0 : disable	0 : disable 1 : enable
37	Transmit Code 39 Check Digit	0x002B	0 : disable	0 : disable 1 : enable
38	Code 39 Full ASCII Conversion	0x0011	0 : disable	0 : disable 1 : enable
39	Code 93	0x0009	1 : enable	0 : disable 1 : enable
40	Set Lengths for Code 93	0x001A	3	0 ~ 255
41		0x001B	30	
42	Code 11	0x000A	1 : enable	0 : disable 1 : enable
43	Set Lengths for Code 11	0x001C	3	0 ~ 255
44		0x001D	30	
45	Code 11 Check Digit Verification	0x0034	0 : Disable	0 : Disable 1 : One Check Digit 2 : Two Check Digits
46	Transmit Code 11 Check Digits	0x002F	0 : Do Not Transmit Code 11 Check Digit(s) (Disable)	0 : Do Not Transmit Code 11 Check Digit(s) (Disable) 1 : Transmit Code 11 Check Digit(s) (Enable)
47	Interleaved 2 of 5 (ITF)	0x0006	0 : disable	0 : disable 1 : enable
48	Set Lengths for Interleaved 2 of 5	0x0016	3	0 ~ 55
49		0x0017	30	
50	I 2 of 5 Check Digit Verification	0x0031	0 : disable	0 : disable 1 : USS Check Digit 2 : OPCC Check Digit
51	Transmit I 2 of 5 Check Digit	0x002C	0 : disable	0 : disable 1 : enable
52	Convert I 2 of 5 to EAN-13	0x0052	0 : disable	0 : disable 1 : enable
53	Discrete 2 of 5 (DTF)	0x0005	1 : enable	0 : disable 1 : enable
54	Set Lengths for Discrete 2 of 5	0x0014	3	0 ~ 55
55		0x0015	30	
56	Codabar (NW - 7)	0x0007	1 : enable	0 : disable 1 : enable
57	Set Lengths for Codabar	0x0018	3	0 ~ 255
58		0x0019	30	
59	CLSI Editing	0x0036	0 : disable	0 : disable 1 : enable
60	NOTIS Editing	0x0037	0 : disable	0 : disable 1 : enable
61	MSI	0x000B	1 : enable	0 : disable 1 : enable
62	Set Lengths for MSI	0x001E	3	0 ~ 255
63		0x001F	30	
64	MSI Check Digits	0x0032	0 : One MSI Check Digit	0 : One MSI Check Digit 1 : Two MSI Check Digits

No.	Param Name	SB Param value	Default setting value	Setting value range
65	Transmit MSI Check Digit(s)	0x002E	0 : Do Not Transmit MSI Check Digit(s) (Disable)	0 : Do Not Transmit MSI Check Digit(s) (Disable) 1 : Transmit MSI Check Digit(s) (Enable)
66	MSI Check Digit Algorithm	0x0033	1 : MOD 10/MOD 10	0 : MOD 10/MOD 11 1 : MOD 10/MOD 10
67	Chinese 2 of 5	0xF098	1 : enable	0 : disable 1 : enable
68	Matrix 2 of 5	0xF16A	1 : enable	0 : disable 1 : enable
69	Set Lengths for Matrix 2 of 5	0xF16B	3	0 ~ 255
70		0xF16C	30	
71	Matrix 2 of 5 Check Digit	0xF16E	0 : disable	0 : disable 1 : enable
72	Transmit Matrix 2 of 5 Check Digit	0xF16F	0 : disable	0 : disable 1 : enable
73	Korean 3 of 5	0xF145	1 : enable	0 : disable 1 : enable
74	Inverse 1D	0xF14A	0 : Regular	0 : Regular 1 : Inverse Only 2 : Inverse Autodetect
75	US Postnet	0x0059	1 : enable	0 : disable 1 : enable
76	US Planet	0x005A	1 : enable	0 : disable 1 : enable
77	Transmit US Postal Check Digit	0x005F	1 : enable	0 : disable 1 : enable
78	UK Postal	0x005B	1 : enable	0 : disable 1 : enable
79	Transmit UK Postal Check Digit	0x0060	1 : enable	0 : disable 1 : enable
80	Japan Postal	0xF022	1 : enable	0 : disable 1 : enable
81	Australia Post	0xF023	1 : enable	0 : disable 1 : enable
82	Australia Post Format	0xF1CE	0 : Autodiscriminate	0 : Autodiscriminate 1 : Raw Format 2 : Alphanumeric Encoding 3 : Numeric Encoding
83	Netherlands KIX Code	0xF046	1 : enable	0 : disable 1 : enable
84	USPS 4CB/One Code/Intelligent Mail	0xF150	1 : enable	0 : disable 1 : enable
85	UPU FICS Postal	0xF163	1 : enable	0 : disable 1 : enable
86	GS1 DataBar	0xF052	1 : enable	0 : disable 1 : enable
87	GS1 DataBar Limited	0xF053	0 : disable	0 : disable 1 : enable
88	GS1 DataBar Limited Security Level	0xF1D8	3 : Security Level 3	1 : Security Level 1 2 : Security Level 2 3 : Security Level 3 4 : Security Level 4
89	GS1 DataBar Expanded	0xF054	0 : disable	0 : disable 1 : enable
90	Convert GS1 DataBar to UPC/EAN	0xF08D	0 : disable	0 : disable 1 : enable
91	Composite CC-C	0xF055	0 : disable	0 : disable 1 : enable
92	Composite CC-A/B	0xF056	0 : disable	0 : disable 1 : enable
93	Composite TLC-39	0xF073	0 : disable	0 : disable 1 : enable
94	UPC Composite Mode	0xF058	1 : UPC Always Linked	0 : UPC Never Linked 1 : UPC Always Linked 2 : Autodiscriminate UPC Composites
95	GS1-128 Emulation Mode for UCC/EAN Composite Codes	0xF0AB	0 : disable	0 : disable 1 : enable

RFID SDK

No.	Param Name	SB Param value	Default setting value	Setting value range
96	PDF417	0x000F	1 : enable	0 : disable 1 : enable
97	MicroPDF417	0x00E3	1 : enable	0 : disable 1 : enable
98	Data Matrix	0xF024	1 : enable	0 : disable 1 : enable
99	Data Matrix Inverse	0xF14C	0 : Regular	0 : Regular 1 : Inverse Only 2 : Inverse Autodetect
100	Decode Mirror Images (Data Matrix Only)	0xF119	0 : Never	0 : Never 1 : Always 2 : Auto
101	Maxicode	0xF026	1 : enable	0 : disable 1 : enable
102	QR Code	0xF025	1 : enable	0 : disable 1 : enable
103	MicroQR	0xF13D	1 : enable	0 : disable 1 : enable
104	Aztec	0xF13E	1 : enable	0 : disable 1 : enable
105	Aztec Inverse	0xF14D	0 : Regular	0 : Regular 1 : Inverse Only 2 : Inverse Autodetect
106	Han Xin	0xF8048F	1 : enable	0 : disable 1 : enable
107	Han Xin Inverse	0xF80490	0 : Regular	0 : Regular 1 : Inverse Only 2 : Inverse Autodetect
108	Redundancy Level	0x004E	1 : Redundancy level 1	1 : Redundancy level 1 2 : Redundancy level 2 3 : Redundancy level 3 4 : Redundancy level 4
109	Security Level	0x004D	1 : Security level 1	0 : Security level 0 1 : Security level 1 2 : Security level 2 3 : Security level 3
110	Intercharacter Gap Size	0xF07D	06 : Normal Intercharacte	06 : Normal Intercharacter Gaps 0A : Large Intercharacter Gaps
111	Decode Session Timeout	0x0088	30	5 ~ 99 (ex 3sec = 30(integer). 5.3sec = 53(integer)
112	Timeout Between Decodes, Same Symbol	0x0089	6	0 ~ 99 (ex 5sec = 50(integer). 5.3sec = 53(integer)
113	Decode Aiming Pattern	0xF032	2 : enable	0 : disable 2 : enable
114	Decoding Illumination	0xF02A	1 : enable	0 : disable 1 : enable
115	Picklist Mode	0xF092	0 : disable	0 : disable 2 : enable

8) Selection Criterias

Selection Criterias : Class that used in selection API's arguments.

■ SelectionCriterias's Criteria Constants

```
public static final int CRITERIA_MIN_COUNT = 1;

public static final int CRITERIA_MAX_COUNT = 8;

public static final int MASK_MAX_SIZE = 64;
```

■ SelectionCriterias Memory Type

: SCMemType class

```
public static class SCMemType {
    public static final int EPC = 1;

    public static final int TID = 2;

    public static final int USER = 3;
}
```

■ SelectionCriterias Action Type

: SCActionType class

```
public static class SCActionType {
    /* Match - Assert SL or inventoried -> A */
    /* Non-Match - Deassert SL or inventoried -> B */
    public static final short ASLINVA_DSLINVB   = 0;

    /* Match - Assert SL or inventoried -> A */
    /* Non-Match - Nothing */
    public static final short ASLINVA_NOTHING   = 1;

    /* Match - Nothing */
    /* Non-Match - Deassert SL or inventoried -> B */
    public static final short NOTHING_DSLINVB   = 2;

    /* Match - Negate SL or (A -> B, B -> A) */
    /* Non-Match - Nothing */
    public static final short NSLINVS_NOTHING   = 3;

    /* Match - Deassert SL or inventoried -> B */
    /* Non-Match - Assert SL or inventoried -> A */
    public static final short DSLINVB_ASLINVA   = 4;

    /* Match - Deassert SL or inventoried -> B */
    /* Non-Match - Nothing */
    public static final short DSLINVB_NOTHING   = 5;

    /* Match - Nothing */
    /* Non-Match - Assert SL or inventoried -> A */
    public static final short NOTHING_ASLINVA   = 6;

    /* Match - Nothing */
    /* Non-Match - Negate SL or (A -> B, B -> A) */
    public static final short NOTHING_NSLINVS   = 7;
}
```

The partitioning of tags into disjoint groups is accomplished by applying actions to the tags that match and/or do not match the specified mask. A selection action is specified using the following:

[action]

Specifies the action that will be applied to the tag populations

(i.e, the matching and non-matching tags).

■ SelectionCriterias Result

: Result class

```
public static class Result {
    public static final int MASK_LENGTH_BIT_ERROR = -6;

    public static final int START_POS_ERROR = -5;

    public static final int MASK_ERROR = -4;

    public static final int ACTION_ERROR = -3;

    public static final int MEMTYPE_ERROR = -2;

    public static final int CRITERIA_COUNT_ERROR = -1;

    public static final int SUCCESS = 0;
}
```

■ SelectionCriterias constructor

: SelectionCriterias()

■ getCriteria value

: getCriteria class

- return ArrayList with criteria values

■ makeCriteria API

	makeCriteria
Declare	**public int makeCriteria(int scMemType, String mask,** **int selectStartPos, int selectMaskLengthBit, int scActionType)**
Description	makeCriteria for Selection API
Parameter	**scMemType** - The memory bank type (EPC = 1, TID = 2, USER = 3) **mask** - HEX format(ex. "3000", "1234ABFF") ~~selectStartPosByte~~ **selectStartPos** - Position that start select is multiply twice value on byte unit **selectMaskLengthBit** - Length of the selected mask(bit)

	scActionType - ASLINVA_DSLINVB = 0 - ASLINVA_NOTHING = 1 - NOTHING_DSLINVB = 2 - NSLINVS_NOTHING = 3 - DSLINVB_ASLINVA = 4 - DSLINVB_NOTHING = 5 - NOTHING_ASLINVA = 6 - NOTHING_NSLINVS = 7
Return	**Success :** ***SelectionCriterias.Result.SUCCESS*** = 0 **Criteria list Error** ***SelectionCriterias.Result.CRITERIA_COUNT_ERROR*** = -1 **Memory Type Error** ***SelectionCriterias.Result.MEMTYPE_ERROR*** = -2 **Action Type Error** ***SelectionCriterias.Result.ACTION_ERROR*** = -3 **Mask Error** ***SelectionCriterias.Result.MASK_ERROR*** = -4 **Start Position Error** ***SelectionCriterias.Result.START_POS_ERROR*** = -5 **Mask Length bit Error** ***SelectionCriterias.Result.MASK_LENGTH_BIT_ERROR*** = -6
Remark	**※Reference 3.6 (Selection Criterias)**

- **Criteria class**

: getSelectMemType()

- Get Criteria's MemType value, return bank value.

: getSelectMask()

- Get Criteria's mask value, return mask value.

: getSelectStartPosByte()

- Get Criteria's start position value, return start position value.

: getSelectMaskLengthBit()

- Get Criteria's mask length bit value, return mask length bit value.

: getSelectAction()

- Get Criteria's action value, return action value.

9) Global Region

1. RFR900/RFR901/HF550XR has 9 or 10 types of Serial Number, and available setting region will be different on each type.

- RFR900Wxxx /RFR901Wxxx/HF550XRWxxx(EU)

Available setting regions are EU(ETSI), India, Iran, Jordan, Pakistan, ~~Morocco,~~ Russia, Cambodia, Myanmar. Cannot set other regions.

Each region will support below country

Region	Nation
EU(ETSI)	Armenia
	Austria
	Azerbaijan
	Belarus
	Belgium
	Bosnia and Herzegovina
	Bulgaria
	Croatia
	Cyprus
	Czech Republic
	Denmark
	Estonia
	Finland
	France
	Germany
	Greece
	Hungary
	Iceland
	Ireland
	Italy
	Latvia
	Lithuania
	Luxembourg
	Macedonia
	Malta
	Moldova
	Netherlands
	Nigeria
	Norway
	Oman
	Poland
	Portugal
	Romania
	Saudi Arabia
	Serbia
	Slovak Republic
	Slovenia
	Spain
	Sweden

	Switzerland
	Tunisia
	Turkey
	United Arab Emirate
	United Kingdom
India	India
Iran	Iran
Jordan	Jordan
Pakistan	Pakistan
~~Morocco~~	~~Morocco~~
Russia	Russia

- RFR900Nxxx/RFR901Nxxx/HF550XRNxxx (FCC)

Available setting regions are FCC, ~~Algeria, Israel,~~ Australia, Bangladesh, Brazil, Brunei, Indonesia, Hongkong, Singapore, Thailand, Vietnam, Korea, Malaysia, New Zealand, Peru, Philippines, South Africa, Uruguay, Taiwan, Venezuela, Guatemala, Macao, Nicaragua.

Cannot set other regions.

Each region will support below country

Region	Nation
FCC	Argentina
	Canada
	Chile
	Colombia
	Costa Rica
	Dominican
	Mexico
	Panama
	United State
	Uruguay
~~Algeria~~	~~Algeria~~
Australia	Australia
Bangladesh	Bangladesh
Brazil	Brazil
Brunei	Brunei
Indonesia	Indonesia
Hongkong	Hongkong
Singapore	Singapore
Thailand	Thailand
Vietnam	Vietnam
Guatemala	Guatemala

Korea	Korea
Malaysia	Malaysia
New Zealand	New Zealand
Peru	Peru
Philippines	Philippines
South Africa	South Africa
Uruguay	Uruguay
Taiwan	Taiwan
Venezuela	Venezuela
Macao	Macao
Nicaragua	Nicaragua

- RFR900Cxxx /RFR901Cxxx/HF550XRCxxx (CH)

Available setting region is China.

Cannot set other regions.

This region will support below country

Region	Nation
China	China

- RFR900J1xxx/RFR901J1xxx (JP)

Available setting region is Japan_1(1W).

Cannot set other regions.

This region will support below country

Region	Nation
Japan_1(1W)	Japan

- RFR900J2xxx/RFR901J2xxx/HF550XRJ2xxx (JP)

Available setting region is Japan_2(250mW).

Cannot set other regions.

This region will support below country

Region	Nation
Japan_2 (250mW)	Japan

- RFR900DZxxx/RFR901DZxxx/HF550XRDZxxx (Algeria)

Available setting region is Algeria

Cannot set other regions.

This region will support below country

Region	Nation

Algeria	Algeria

- RFR900MAxxx/RFR901MAxxx/HF550XRMAxxx (Morocco)

Available setting region is Morocco

Cannot set other regions.

This region will support below country

Region	Nation
Morocco	Morocco

- RFR900EGxxx/RFR901EGxxx/HF550XREGxxx (Egypt)

Available setting region is Egypt

Cannot set other regions.

This region will support below country

Region	Nation
Egypt	Egypt

- RFR900CLxxx/RFR901CLxxx/HF550XRCLxxx (Chile)

Available setting region is Chile

Cannot set other regions.

This region will support below country

Region	Nation
Chile	Chile

- RFR900ILxxx /RFR901ILxxx/HF550XRILxxx (Israel)

Available setting region is Israel

Cannot set other regions.

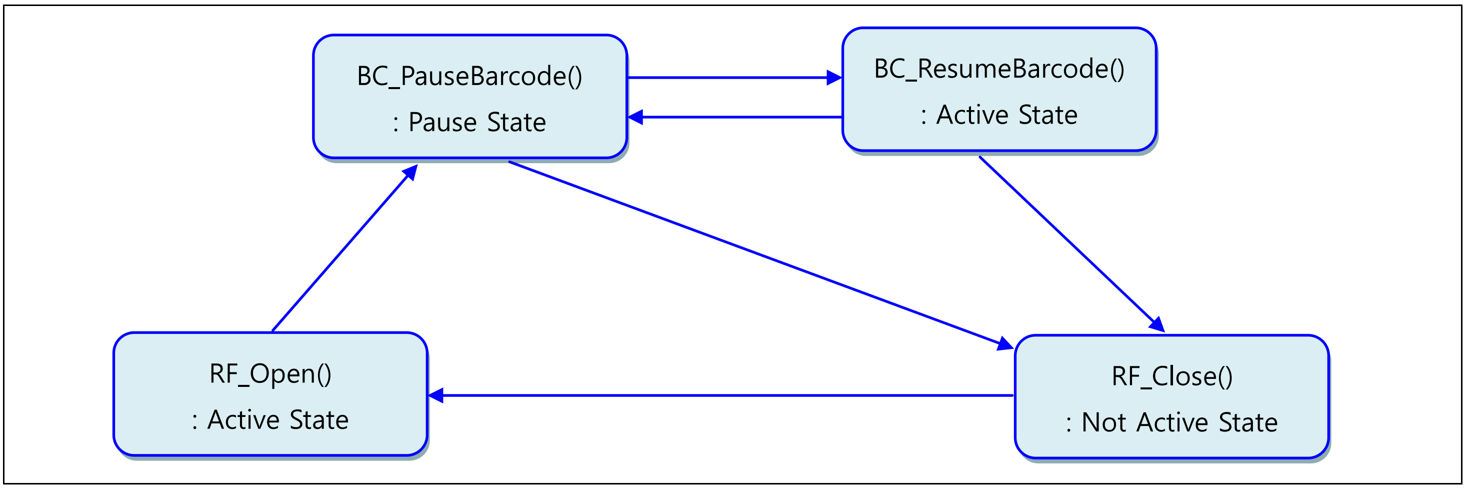
This region will support below country

Region	Nation
Israel	Israel

10) BC Barcode Lifecycle

Barcode Lifecycle in SLED Library

: Reference for BC_PauseBarcode / BC_ResumeBarcode APIs.



11) Bluetooth callback message (BTReader)

■ SLED_BT_DEVICE_FOUND

This callback message occur when SLED found device after scanning and always contains bundle data.

Bundle data	Key	Data format
Name	ConstantsBT.BT_BUNDLE_NAME_KEY	String
Address	ConstantsBT.BT_BUNDLE_ADDR_KEY	String
Bond state	ConstantsBT.BT_BUNDLE_BOND_STATE_KEY	Int

■ SLED_BT_BOND_STATE_CHANGED

This callback message occur when Pairing status change and always contains bundle data.

Bundle data	Key	Data format
Name	ConstantsBT.BT_BUNDLE_NAME_KEY	String
Address	ConstantsBT.BT_BUNDLE_ADDR_KEY	String
Bond state	ConstantsBT.BT_BUNDLE_BOND_STATE_KEY	Int
Bond new state	ConstantsBT.BT_BUNDLE_BOND_NEW_STATE_KEY	Int
Bond previous state	ConstantsBT.BT_BUNDLE_BOND_PREV_STATE_KEY	Int

■ SLED_BT_ACL_CONNECTED

This callback message occur when Changing to a connected state and always contains bundle data.

Bundle data	Key	Data format
Name	ConstantsBT.BT_BUNDLE_NAME_KEY	String
Address	ConstantsBT.BT_BUNDLE_ADDR_KEY	String
Bond state	ConstantsBT.BT_BUNDLE_BOND_STATE_KEY	Int

- SLED_BT_ACL_DISCONNECT_REQUESTED

This callback message occur when Receive disconnect request and always contains bundle data.

Bundle data	Key	Data format
Name	ConstantsBT.BT_BUNDLE_NAME_KEY	String
Address	ConstantsBT.BT_BUNDLE_ADDR_KEY	String
Bond state	ConstantsBT.BT_BUNDLE_BOND_STATE_KEY	Int

- SLED_BT_ACL_DISCONNECTED

This callback message occur when Changing to a disconnect state and always contains bundle data.

Bundle data	Key	Data format
Name	ConstantsBT.BT_BUNDLE_NAME_KEY	String
Address	ConstantsBT.BT_BUNDLE_ADDR_KEY	String
Bond state	ConstantsBT.BT_BUNDLE_BOND_STATE_KEY	Int

- SLED_BT_STATE_CHANGED

This callback message occur when Changed device's Bluetooth state and always contains bundle data.

Bundle data	Key	Data format
Bond new state	ConstantsBT.BT_BUNDLE_BOND_NEW_STATE_KEY	Int
Bond previous state	ConstantsBT.BT_BUNDLE_BOND_PREV_STATE_KEY	Int

- SLED_BT_DISCOVERY_STARTED

This callback message occur when Start the scan and not contains bundle data.

- SLED_BT_DISCOVERY_FINISHED

This callback message occur when The scan complete and not contains bundle data

- SLED_BT_PAIRING_REQUEST

This callback message occur when Receive pair request and not contains bundle data.

12) APIs

Reader	
Declare	**Public static synchronized Reader getReader** **(Context context, Handler handler)**
Description	Gets the instance of reader
Parameter	Context Handler

Return	Reader
Remark	**Serial Reader**

	BTReader
Declare	**Public static synchronized BTReader getReader** **(Context context, Handler handler)**
Description	Gets the instance of Bluetooth reader
Parameter	Context Handler
Return	Reader
Remark	**Bluetooth Reader** **[Requires permission]** - **android.Manifest.permission.BLUETOOTH**

■ RF APIs

RF_GetDutyCycle		
Declare		**public int RF_GetDutyCycle()**
Description		Gets the duty cycle value of the RFID radio module
Parameter		Void
Return	**Reader**	**Success :** Value of the Duty Cycle **(MIN_DUTY(0) ~ MAX_DUTY(1000))** **Serial Error** SDConsts.RFDutyCycle.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFDutyCycle.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Connected Error** SDConsts.RFDutyCycle.***SD_NOT_CONNECTED*** = -5 **Other Error** SDConsts.RFDutyCycle.***OTHER_ERROR*** = -1 *** Can receive other error constant of "RFDutyCycle" class.**
	BTReader	**Success :** Value of the Duty Cycle **(MIN_DUTY(0) ~ MAX_DUTY(1000))** **Enabled Error** : SDConsts.RFDutyCycle.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFDutyCycle.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFDutyCycle.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFDutyCycle.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Hotswap Error** : SDConsts.RFDutyCycle.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFDutyCycle" class.**
Remark		**※ Reference (3.2.RFDutyCycle)** **※ [(BTReader)Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_SetDutyCycle		
Declare		**Public int RF_SetDutyCycle(int millisec)**
Description		Sets the duty cycle value of the RFID radio module
Parameter		**millisec(0 ~ 1000) : 100(default)** – Import or set the resting time(millisecond) of each ports.
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 **Range Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0

		Enabled Error : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		RF_GetAccessTimeout
Declare		**public int RF_GetAccessTimeout()**
Description		Gets the timeout value of access apis(ref. IRfidAccess interface) for the RFID radio module
Parameter		Void
Return	**Reader**	**Success :** Value of the AccessTimeout **(MIN_ACCESS_TIMEOUT(100) ~ MAX_ACCESS_TIMEOUT(10000))** **Serial Error** SDConsts.RFAccessTimeout.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts. RFAccessTimeout.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Connected Error** SDConsts. RFAccessTimeout.***SD_NOT_CONNECTED*** = -5 **Other Error** SDConsts. RFAccessTimeout.***OTHER_ERROR*** = -1 *** Can receive other error constant of "RFAccessTimeout" class.**
	BTReader	**Success :** Value of the AccessTimeout **(MIN_ACCESS_TIMEOUT(100) ~ MAX_ACCESS_TIMEOUT(10000))** **Enabled Error** : SDConsts.RFAccessTimeout.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFAccessTimeout.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFAccessTimeout.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFAccessTimeout.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Hotswap Error** : SDConsts.RFAccessTimeout.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFAccessTimeout" class.**
Remark		**※ Reference (3.2.RFAccessTimeout)** **※ [(BTReader)Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		RF_SetAccessTimeout
Declare		**Public int RF_SetAccessTimeout(int millisec)**
Description		Sets the timeout value of access apis(ref. IRfidAccess interface) for the RFID radio module.

Parameter		**millisec(100 ~ 10000) : 3000(default)** – Timeout value(millisecond) of access apis
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 **Range Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_GetRadioPowerState

Declare		**public int RF_GetRadioPowerState()**
Description		Gets the power state value of the RFID radio module
Parameter		void
Return	**Reader**	**Success :** Value of the Power State **(MIN_POWER(5) ~ MAX_POWER(30))** **Serial Error** SDConsts.RFPower.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFPower.***READER_OR_SERIAL_STATUS_ERROR***= -7 **Command State Error** SDConsts.RFPower.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFPower.***SD_NOT_CONNECTED***= -5 *** Can receive other error constant of "RFPower" class.**
	BTReader	**Success :** Value of the Power State **(MIN_POWER(5) ~ MAX_POWER(30))** **Enabled Error** : SDConsts.RFPower.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFPower.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFPower.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFPower.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFPower.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFPower.***ERROR_HOTSWAP_STATE*** = -37

		*** Can receive other error constant of “RFPower” class.**
Remark		**※ Reference (3.2.RFPower)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		RF_SetRadioPowerState
Declare		**Public int RF_SetRadioPowerState(int RFPower)**
Description		Sets the power state value of the RFID radio module
Parameter		**RFPower** : The power level for the antenna port **(5 ~ 30) : 30(default)** - 30dbm for 30, 29dBm for 29, …
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 **Range Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of “RFResult” class.**
	BTReader	**Success :** Constants.RFResult.***SUCCESS*** = 0 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of “RFResult” class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH** **※ notice for setting power** **- In Algeria and Egypt, the power limit is only 100mW, so user can set the power to only 17** **- In Chile, the power limit is only 100mW, so user can set the power to only 17** **- In Morocco, the power limit is only 500mW, so user can set the power to only 26** **- If you set a larger value, it will only be set to th actual maximum value** **(Ex. Algeria, Egypt = 17 / Morocco = 26)**

RF_GetRFMode

Declare		**public int RF_GetRFMode()**
Description		Gets the RFMode(link profile) value of the RFID radio module
Parameter		Void
Return	**Reader**	**Success :** Value of the RF Mode **(DSB_ASK_1(0) ~ DSB_ASK_2(3))** **Serial Error** SDConsts.RFMode.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFMode.***READER_OR_SERIAL_STATUS_ERROR***= -7 **Command State Error** SDConsts.RFMode.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFMode.***SD_NOT_CONNECTED***= -5 *** Can receive other error constant of "RFMode" class.**
	BTReader	**Success :** Value of the RF Mode **(DSB_ASK_1(0) ~ DSB_ASK_2(3))** **Enabled Error** : SDConsts.RFMode.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFMode.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFMode.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFMode.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFMode.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFMode.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFMode" class.**
Remark		**※ Reference (3.2.RFMode)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_SetRFMode

Declare		**public int RF_SetRFMode(int RFMode)**
Description		Sets the RFMode(link profile) value of the RFID radio module.
Parameter		**RFMode** **:** Link Profile **(0 ~ 3) : 1(default)**

Profile Index	**0**	**1**	**2**	**3**
R-T Modulation	DSB-ASK	PR-ASK	PR-ASK	DSB-ASK
Tari (us)	25.00	25.00	25.00	6.25
T-R Modulation	FM0	Miller-4	Miller-4	FM0
LF (kHz)	40.00	250.00	300.00	400.00

Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0

		Range Error SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success :** Constants.RFResult.***SUCCESS*** = 0 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_GetSingulationControl

Declare		**public int RF_GetSingulationControl()**
Description		Gets the singulation value of RFID radio module
Parameter		void
Return	**Reader**	**Success :** Value of the Singulation **(MIN_SINGULATION(0) ~ MAX_SINGULATION(15))** The starting Q value to use. Valid values are 0~15, inclusive. startQValue must be greater than or equal to minimumQValue and less than or equal to maximumQValue. **Serial Error** SDConsts.RFSingulation.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFSingulation.***READER_OR_SERIAL_STATUS_ERROR***= -7 **Command State Error** SDConsts.RFSingulation.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFSingulation.***SD_NOT_CONNECTED***= -5 *** Can receive other error constant of "RFSingulation" class.**
	BTReader	**Success :** Value of the Singulation **(MIN_SINGULATION(0) ~ MAX_SINGULATION(15))** The starting Q value to use. Valid values are 0~15, inclusive. startQValue must be greater than or equal to minimumQValue and less than or equal to maximumQValue.

		Enabled Error : SDConsts.RFSingulation.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFSingulation.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFSingulation.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFSingulation.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFSingulation.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFSingulation.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFSingulation" class.**
Remark		**※ Reference (3.2.RFSingulation)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_GetMinSingulationControl		
Declare		**public int RF_GetMinSingulationControl()**
Description		Gets minimum singulation value of RFID radio module
Parameter		void
Return	**Reader**	**Success :** Value of the minimum singulation **Serial Error** SDConsts.RFSingulation.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFSingulation.***READER_OR_SERIAL_STATUS_ERROR***= -7 **Command State Error** SDConsts.RFSingulation.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFSingulation.***SD_NOT_CONNECTED***= -5 *** Can receive other error constant of "RFSingulation" class.**
	BTReader	**Success :** Value of the minimum singulation **Enabled Error** : SDConsts.RFSingulation.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFSingulation.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFSingulation.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFSingulation.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFSingulation.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFSingulation.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFSingulation" class.**
Remark		**※ Reference (3.2.RFSingulation)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_GetMaxSingulationControl		
Declare		**public int RF_GetMaxSingulationControl()**
Description		Gets the maximum singulation value of RFID radio module

Parameter		void
Return	**Reader**	**Success :** Value of the maximum singulation **Serial Error** SDConsts.RFSingulation.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFSingulation.***READER_OR_SERIAL_STATUS_ERROR***= -7 **Command State Error** SDConsts.RFSingulation.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFSingulation.***SD_NOT_CONNECTED***= -5 *** Can receive other error constant of "RFSingulation" class.**
	BTReader	**Success :** Value of the maximum singulation **Serial Error** SDConsts.RFSingulation.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFSingulation.***READER_OR_SERIAL_STATUS_ERROR***= -7 **Command State Error** SDConsts.RFSingulation.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFSingulation.***SD_NOT_CONNECTED***= -5 **Hotswap Error** : SDConsts.RFSingulation.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFSingulation" class.**
Remark		**※ Reference (3.2.RFSingulation)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		RF_SetSingulationControl
Declare		**public int RF_SetSingulationControl(int RFSingulation,** **int minSingulation, int maxSingulation)**
Description		Sets the minimum singulation, singulation, maximum singulation values of RFID radio module
Parameter		**RFSingulation (0 ~ 15) : 4(default)** - Start Q : Singulation Algorithm DynamicQ **minSingulation** - Minimum value of singulation range **maxSingulation** - Maximum value of singulation range
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 **Range Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFResult" class.**

	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		RF_ResetConfigToFactoryDefaults
Declare		**public int RF_ResetConfigToFactoryDefaults()**
Description		Resets the setting values of RFID radio module
Parameter		void
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 **[※ Setting with default Values]** **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Other Error** SDConsts.RFResult.***OTHER_ERROR*** = -1 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 **[※ Setting with default Values]** **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Other Error** : SDConsts.RFResult.***OTHER_ERROR*** = -14 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ Reference 3.2 (Default)** **- (3.2) Default** **※ [(BTReader) Requires permission]**

		- **android.Manifest.permission.BLUETOOTH**

RF_GetRegion		
Declare		**public int RF_GetRegion()**
Description		Gets the region value of RFID radio module (ETSI, FCC, etc)
Parameter		void
Return	**Reader**	**Success :** Value of the Region **(KOREA(0) ~ CHILE(32), UNKNOWN(-1))** **Serial Error** SDConsts.RFRegion.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFRegion.***READER_OR_SERIAL_STATUS_ERROR***= -7 **Command State Error** SDConsts.RFRegion.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFRegion.***SD_NOT_CONNECTED***= -5 *** Can receive other error constant of "RFRegion" class.**
	BTReader	**Success :** Value of the Region **(KOREA(0) ~ CHILE(32), UNKNOWN(-1))** **Enabled Error** : SDConsts.RFRegion.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFRegion.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFRegion.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFRegion.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFRegion.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFRegion.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFRegion" class.**
Remark		**※ Reference (3.2.RFRegion)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_SetRegion		
Declare		**public int RF_SetRegion(int RFRegion)**
Description		Sets the Region value of RFID radio module
Parameter		**RFRegion (0 ~ 32)** - Import or sets-up history of country-by-country frequency. **UNKNOWN = -1, KOREA = 0, ETSI = 1, FCC = 2,** **AUSTRALIA = 3, BANGLADESH = 4, BRAZIL = 5, BRUNEI = 6,** **CHINA = 7, HONGKONG = 8, INDIA = 9, INDONESIA = 10,** **IRAN = 11, ISRAEL = 12, JAPAN_1 = 13, JAPAN_2 = 14,** **JORDAN = 15, MALAYSIA = 16, MOROCCO = 17, NEW_ZEALAND = 18,** **PAKISTAN = 19, PERU = 20,**

		PHILIPPINES = 21, SINGAPORE = 22, SOUTH_AFRICA = 23, **TAIWAN = 24, THAILAND = 25, URUGUAY = 26,** **VENEZUELA = 27, VIETNAM = 28, RUSSIA = 29, ALGERIA = 30, EGYPT = 31,** **CHILE = 32**
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - REGION_CHANGE_START = 21 - REGION_CHANGE_END = 22 **Range Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Region Error** SDConsts.RFResult.***OTHER_ERROR*** = -1 (For a region that does not support will return Region Error after check serial number) *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - REGION_CHANGE_START = 21 - REGION_CHANGE_END = 22 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Other Error** : SDConsts.RFResult.***OTHER_ERROR*** = -1 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 (For a region that does not support will return Region Error after check serial number) *** Can receive other error constant of "RFResult" class.**
Remark		**In case of this API, Run time during about 0 ~ 8 seconds is required.** **It sends related callback message *(REGION_CHANGE_START(21) → REGION_CHANGE_END(22))* at the beginning and the end.** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_GetAvailableRegionAtThisDevice		
Declare		**public String RF_GetAvailableRegionAtThisDevice()**
Description		Gets the available region value at this sled device
Parameter		void
Return	**Reader**	**Success :** Available region string (Ex. “RFRegion:ETSI=1; INDIA=9; IRAN=11; JORDAN=15; PAKISTAN=19; RUSSIA=29;) **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = “-4” **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = “-7” **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = “-4” **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = “-5” **Region Unknown Error** “RFRegion:UNKNOWN=-1;”
	BTReader	**Success :** Available region string (Ex. “RFRegion:ETSI=1; INDIA=9; IRAN=11; JORDAN=15; PAKISTAN=19; RUSSIA=29;) **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of “RFResult” class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

~~**RF_SetRegionAuto**~~	
~~**Declare**~~	~~**public String RF_SetRegionAuto()**~~
~~**Description**~~	~~Set the Region for the reader automatically~~
~~**Parameter**~~	~~void~~
~~**Reader Return**~~	~~**Success :** SDConsts.RFResult.***SUCCESS*** = 0~~ ~~**Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = “-4”~~ ~~**Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = “-7”~~ ~~**Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = “-4”~~ ~~**Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = “-5”~~ ~~**Other Error** SDConsts.RFResult.***OTHER_ERROR*** = -1~~
~~**Remark**~~	**※ This API is deprecated**

		RF_GetLibVersion
Declare		**public String RF_GetLibVersion()**
Description		Gets the version information of the SLED library(jar).
Parameter		void
Return	**Reader**	**Success :** Value of the Library Version **Serial Error** SDConsts.***ERROR_STR*** = "Error" **Condition Error** SDConsts.***ERROR_STR*** = "Error" **Connected Error** SDConsts.***ERROR_STR*** = "Error"
	BTReader	**Success :** Value of the Library Version **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : "Error" **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

	~~**RF_Open**~~
~~**Declare**~~	~~**public boolean RF_Open()**~~
~~**Description**~~	~~Ready for all communication(Serial, Barcode and so on)~~
~~**Parameter**~~	~~void~~
~~**Reader Return**~~	~~**Success :** True~~ ~~**Fail :** False~~
~~**Remark**~~	~~**RF_Open API has function of barcode open**~~ **※ This API is deprecated, use SD_Open**

	~~**RF_Open**~~
~~**Declare**~~	~~**public boolean RF_Open(String clientId)**~~
~~**Description**~~	~~Ready for all communication with specific client feature.~~ ~~(Serial, Barcode and so on)~~
~~**Parameter**~~	~~clientId~~

	~~- Company id String~~
~~**Reader Return**~~	~~**Success :** True~~ ~~**Fail :** False~~
~~**Remark**~~	~~**RF_Open API has function of barcode open**~~ **※ This API is deprecated, use SD_Open(String clientId)**

	~~**RF_Close**~~
~~**Declare**~~	~~**public boolean RF_Close()**~~
~~**Description**~~	~~Close all opened communication(Serial, Barcode and so on)~~
~~**Parameter**~~	~~void~~
~~**Reader Return**~~	~~**Success :** True~~ ~~**Fail :** False~~
~~**Remark**~~	~~**RF_Close API has function of barcode close**~~ **※ This API is deprecated, use SD_Close()**

		RF_GetRssiTrackingState
Declare		**public int RF_GetRssiTrackingState()**
Description		Gets the state of RSSI Tracking
Parameter		void
Return	**Reader**	**Success :** Value of RSSI Tracking State **(Off(0), On(1))** **Serial Error** SDConsts.RFRssi.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFRssi.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Connected Error** SDConsts.RFRssi.***SD_NOT_CONNECTED*** = -5 **Other Error** SDConsts.RFRssi.***OTHER_ERROR*** = -1 *** Can receive other error constant of "RFRssi" class.**
	BTReader	**Success :** Value of RSSI Tracking State **(Off(0), On(1))** **Enabled Error** : SDConsts.RFRssi.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFRssi.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFRssi.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFRssi.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Other Error** : SDConsts.RFRssi.***OTHER_ERROR*** = -1 **Hotswap Error** : SDConsts.RFRssi.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFRssi" class.**

Remark		※ Reference (3.2.RFRssi) ※ [(BTReader) Requires permission] - android.Manifest.permission.BLUETOOTH

RF_SetRssiTrackingState		
Declare		**public int RF_SetRssiTrackingState(int RFRssi)**
Description		Sets the state of RSSI Tracking
Parameter		**RFRssi (On = 1, Off = 0) : 1(default)** - RSSI value
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 **Range Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of “RFResult” class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Hotswap Error** : SDConsts.RFRssi.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of “RFResult” class.**
Remark		※ [(BTReader) Requires permission] - android.Manifest.permission.BLUETOOTH

RF_GetSession		
Declare		**public int RF_GetSession()**
Description		Gets the session value of the RFID radio module(Session flag will be matched against the inventory state specified by target)
Parameter		void
Return	**Reader**	**Success :** Value of the Session **(SESSION_S0(0) ~ SESSION_S3(3))** **Serial Error** SDConsts.RFSession.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFSession.***READER_OR_SERIAL_STATUS_ERROR*** = -7

		Connected Error SDConsts.RFSession.***SD_NOT_CONNECTED*** = -5 **Other Error** SDConsts.RFSession.***OTHER_ERROR*** = -1 *** Can receive other error constant of "RFSession" class.**
	BTReader	**Success :** Value of the Session **(SESSION_S0(0) ~ SESSION_S3(3))** **Enabled Error** : SDConsts.RFSession.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFSession.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFSession.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFSession.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Other Error** : SDConsts.RFSession.***OTHER_ERROR*** = -1 **Hotswap Error** : SDConsts.RFSession.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFSession" class.**
Remark		**Only operate when the toggle state is OFF.** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		RF_SetSession
Declare		**public int RF_SetSession(int RFSession)**
Description		Sets the session value of the RFID radio module(Session flag will be matched against the inventory state specified by target)
Parameter		**RFSession (0~3) : 0(default)** - Value of the Session
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 **Range Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**Only operate when the toggle state is OFF.** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_GetToggle

Declare		**public int RF_GetToggle()**
Description		Gets the toggle state of the RFID radio module (A flag that indicates, after performing the inventory cycle for the specified target, if the target should be toggled and another inventory cycle run.)
Parameter		void
Return	**Reader**	**Success :** Value of the toggle state **(Off(0), On(1))** **Serial Error** SDConsts.RFToggle.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFToggle.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFToggle.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFToggle.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFToggle" class.**
	BTReader	**Success :** Value of the toggle state **(Off(0), On(1))** **Enabled Error** : SDConsts.RFToggle.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFToggle.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFToggle.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFToggle.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFToggle.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFToggle.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFToggle" class.**
Remark		**※ Reference (3.2.RFToggle)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_SetToggle

Declare		**public int RF_SetToggle(int RFToggle)**
Description		Sets the toggle state of the RFID radio module (A flag that indicates, after performing the inventory cycle for the specified target, if the target should be toggled and another inventory cycle run.)
Parameter		**RFToggle (0 ~ 1) : 1(default)** - 0 : OFF – should not be toggled - 1 : ON – should be toggled
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0

		Range Error SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of “RFResult” class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of “RFResult” class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_RemoveSelection		
Declare		**public int RF_RemoveSelection()**
Description		Resets the selection values of the RFID radio module
Parameter		void
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 **(RFMemType = 0 / mask = Null / maskStartPos = 0)** **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of “RFResult” class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 **(RFMemType = 0 / mask = Null / maskStartPos = 0)** **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37

		*** Can receive other error constant of "RFResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		RF_SetSelection
Declare		**public int RF_SetSelection(SelectionCriterias selectionCriteria)**
Description		Sets the selection values of RFID radio module
Parameter		**selectionCriteria**
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 **Range Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 **Range Error** : SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ Reference 3.6 (SelectionCriterias)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		RF_GetSelection
Declare		**public SelectionCriterias RF_GetSelection()**
Description		Gets the selection values of RFID radio module
Parameter		void
Return	**Reader**	**Success :** Value of the selection **Other Error** : *NULL*

	BTReader	**Success :** Value of the selection **Other Error** : *Null*
Remark		※ **[(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_ModuleReboot		
Declare		**public int RF_ModuleReboot()**
Description		Reboots RFID module (not SLED)
Parameter		void
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		※ **[(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_GetDwelltime		
Declare		**public int RF_GetDwelltime()**
Description		Gets the dwell time (30 ~ 400, 200(default)) of the RFID radio module
Parameter		void
Return	**Reader**	**Success :** Value of the dwell time **(MIN_DWELL(30)~MAX_DWELL(400)) : 200(default)** **Serial Error** SDConsts.RFDwell.***OTHER_CMD_RUNNING_ERROR*** = -4

		Condition Error SDConsts.RFDwell.***READER_OR_SERIAL_STATUS_ERROR***= -7 **Command State Error** SDConsts.RFDwell.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFDwell.***SD_NOT_CONNECTED***= -5 *** Can receive other error constant of “RFDwell” class.**
	BTReader	**Success :** Value of the dwell time **(MIN_DWELL(30)~MAX_DWELL(400)) : 200(default)** **Enabled Error** : SDConsts.RFDwell.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFDwell.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFDwell.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFDwell.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFDwell.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFDwell.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of “RFDwell” class.**
Remark		**※ Reference (3.2.RFDwell)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_SetDwelltime

Declare		**public int RF_SetDwelltime(int RFDwell)**
Description		Sets the dwell time of RFID radio module
Parameter		**RFDwell (30~400) : 200(default)** - The number of milli iseconds to spend on this antenna port during a cycle
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 **Range Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of “RFResult” class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 **Range Error** : SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4

		Hotswap Error : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_GetRFIDVersion		
Declare		**public String RF_GetRFIDVersion()**
Description		Gets the version of RFID radio module
Parameter		void
Return	**Reader**	**Success :** Value of the RFID Version **Serial Error** SDConsts.***ERROR_STR*** = "Error" **Condition Error** SDConsts.***ERROR_STR*** = "Error" **Command State Error** SDConsts.***ERROR_STR*** = "Error" **Connected Error** SDConsts.***ERROR_STR*** = "Error"
	BTReader	**Success :** Value of the RFID Version **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : "Error" **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_UpdateRFIDFirmware		
Declare		**public int RF_UpdateRFIDFirmware(String filepath)**
Description		Updates firmware of RFID radio module
Parameter		filepath - File path for RFID firmware update
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - UPDATE_RF_FW_START = 23 - UPDATE_RF_FW = 24

		- UPDATE_RF_FW_END = 25 **File path Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - UPDATE_RF_FW_START = 23 - UPDATE_RF_FW = 24 - UPDATE_RF_FW_END = 25 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **File Path Error** : SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Charge Error** : SDConsts.RFResult.***CHARGING_STATE_ERROR*** = -14 *** Can receive other error constant of "RFResult" class.**
Remark		**In case of this API, Run time during about 90 seconds is required.** **It sends related callback message*(UPDATE_RF_FW_START(23) → UPDATE_RF_FW(24) → UPDATE_RF_FW_END(25))* at the beginning and the end.** **If it start FW's update, we recommend that do not be call any other cmd.** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH** - **android.Manifest.permission.WRITE_EXTERNAL_STORAGE** - **android.Manifest.permission.READ_EXTERNAL_STORAGE** **※ [(Reader) Requires permission]** - **android.Manifest.permission.WRITE_EXTERNAL_STORAGE** - **android.Manifest.permission.READ_EXTERNAL_STORAGE**

	RF_UpdateRFIDFirmware
Declare	**public int RF_UpdateRFIDFirmware(Uri uri)**

Description		Updates firmware of RFID radio module
Parameter		uri - File path for RFID firmware update
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - UPDATE_RF_FW_START = 23 - UPDATE_RF_FW = 24 - UPDATE_RF_FW_END = 25 **File path Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - UPDATE_RF_FW_START = 23 - UPDATE_RF_FW = 24 - UPDATE_RF_FW_END = 25 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **File Path Error** : SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Charge Error** : SDConsts.RFResult.***CHARGING_STATE_ERROR*** = -14 *** Can receive other error constant of "RFResult" class.**
Remark		**In case of this API, Run time during about 90 seconds is required.** **It sends related callback message*(UPDATE_RF_FW_START(23) → UPDATE_RF_FW(24) → UPDATE_RF_FW_END(25))* at the beginning and the end.** **If it start FW's update, we recommend that do not be call any other cmd.** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH** - **android.Manifest.permission.WRITE_EXTERNAL_STORAGE** - **android.Manifest.permission.READ_EXTERNAL_STORAGE** **※ [(Reader) Requires permission]**

		- **android.Manifest.permission.WRITE_EXTERNAL_STORAGE** - **android.Manifest.permission.READ_EXTERNAL_STORAGE**

RF_GetInventorySessionTarget

Declare		**public int RF_GetInventorySessionTarget()**
Description		Gets inventory session target of RFID radio module
Parameter		void
Return	**Reader**	**Success :** Value of Inventory session **(TARGET_A(0), TARGET_B(1))** **Serial Error** SDConsts.RFInvSessionTarget.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFInvSessionTarget.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Connected Error** SDConsts.RFInvSessionTarget.***SD_NOT_CONNECTED*** = -5 **Other Error** SDConsts.RFInvSessionTarget.***OTHER_ERROR*** = -1 *** Can receive other error constant of "RFInvSessionTarget" class.**
	BTReader	**Success :** Value of Inventory session **(TARGET_A(0), TARGET_B(1))** **Enabled Error** : SDConsts.RFInvSessionTarget.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFInvSessionTarget.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFInvSessionTarget.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFInvSessionTarget.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFInvSessionTarget.***OTHER_CMD_RUNNING_ERROR*** = -4 **Other Error** : SDConsts.RFInvSessionTarget.***OTHER_ERROR*** = -1 **Hotswap Error** : SDConsts.RFInvSessionTarget.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFInvSessionTarget" class.**
Remark		**※ Reference (3.2.RFInvSessionTarget)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_SetInventorySessionTarget

Declare		**public int RF_SetInventorySessionTarget(int RFInvSessionTarget)**
Description		Sets the inventory session target of the RFID radio module
Parameter		RFInvSessionTarget **(0 ~ 1) : 0(default)** - TARGET_A : 0 - TARGET_B : 1
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 **Range Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3

		Serial Error SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 **Range Error** : SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**Only operate when the toggle state is OFF.** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_GetSelectionFlag

Declare		**public int RF_GetSelectionFlag()**
Description		Gets the selection flag
Parameter		void
Return	**Reader**	**Success** : Value of selection flag **(ALL(1), DEASSERTED(2), ASSERTED(3, default))** **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Other Error** SDConsts.RFResult.***OTHER_ERROR*** = -1 *** Can receive other error constant of "RFSelectionFlag" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Other Error** : SDConsts.RFResult.***OTHER_ERROR*** = -1 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFSelectionFlag" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		RF_SetSelectionFlag
Declare		**public int RF_SetSelectionFlag(int RFSelectionFlag)**
Description		Sets the selection flag Specifies the state of the selected (SL) flag for tags that will have the operation applied to them Only operate when the selection option enabled state
Parameter		void
Return	**Reader**	**Success** : Constants.RFResult.***SUCCESS*** = 0 **Range Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 **Range Error** : SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

	RF_PerformInventory
Declare	**public int RF_PerformInventory(boolean turboMode,** **boolean enableSelection, boolean ignorePC)**
Description	Performs the inventory operation
Parameter	**turboMode : 1(default)** - True : Continuous mode(Duty cycle = 0) - False : Non Continuous mode **enableSelection** - True : Select enable(Set RF_SetSelection API first.) - False : Select disable **ignorePC**

- True : The tag data that removed PC field.
- False : The tag data that included PC field.

Return **Reader**

Success Constants.RFResult.***SUCCESS*** = 0

[Auto-update message from SLED]

SDConsts.RFCmdMsg

- INVENTORY = 5

Serial Error SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4

Condition Error SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7

Command State Error SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4

Connected Error SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5

*** Can receive other error constant of “RFResult” class.**

BTReader

Success Constants.RFResult.***SUCCESS*** = 0

[Auto-update message from SLED]

SDConsts.RFCmdMsg

- INVENTORY = 5

Enabled Error : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15

Connected Error : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5

Block State Error : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4

Condition Error : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7

Command State Error : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4

Mode Error : SDConsts.RFResult.***MODE_ERROR*** = -6

Battery Error : SDConsts.RFResult.***LOW_BATTERY*** = -12

Hotswap Error : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37

*** Can receive other error constant of “RFResult” class.**

Remark

※ [(BTReader) Requires permission]

- **android.Manifest.permission.BLUETOOTH**

※ Optiamal RF configuration values.

	Value
RF Mode	**1**
Sesstion	**S1**
Toggle	**OFF**
Singulation	**10**

RF_PerformInventory

public int RF_PerformInventory(boolean turboMode,
boolean enableSelection, boolean ignorePC, boolean isEPCDecode)

Performs the inventory operation

turboMode : 1(default)

- True : Continuous mode(Duty cycle = 0)
- False : Non Continuous mode

enableSelection

- True : Select enable(Set RF_SetSelection API first.)
- False : Select disable

ignorePC

- True : The tag data that removed PC field.
- False : The tag data that included PC field.

isEPCDecode

- True : Set receive data with EPC decode data.
- False : Set receive data without EPC decode data.

Success Constants.RFResult.*SUCCESS* = 0

[Auto-update message from SLED]

SDConsts.RFCmdMsg

- INVENTORY = 5

Serial Error SDConsts.RFResult.*OTHER_CMD_RUNNING_ERROR* = -4

Condition Error SDConsts.RFResult.*READER_OR_SERIAL_STATUS_ERROR* = -7

Command State Error SDConsts.RFResult.*OTHER_CMD_RUNNING_ERROR* = -4

Connected Error SDConsts.RFResult.*SD_NOT_CONNECTED* = -5

* Can receive other error constant of “RFResult” class.

Success Constants.RFResult.*SUCCESS* = 0

[Auto-update message from SLED]

SDConsts.RFCmdMsg

- INVENTORY = 5

Enabled Error : SDConsts.RFResult.*BLUETOOTH_NOT_ENABLED* = -15

Connected Error : SDConsts.RFResult.*SD_NOT_CONNECTED* = -5

Block State Error : SDConsts.RFResult.*OTHER_CMD_RUNNING_ERROR* = -4

Condition Error : SDConsts.RFResult.*READER_OR_SERIAL_STATUS_ERROR* = -7

Command State Error : SDConsts.RFResult.*OTHER_CMD_RUNNING_ERROR* = -4

Mode Error : SDConsts.RFResult.*MODE_ERROR* = -6

Battery Error : SDConsts.RFResult.*LOW_BATTERY* = -12

Hotswap Error : SDConsts.RFResult.*ERROR_HOTSWAP_STATE* = -37

* Can receive other error constant of “RFResult” class.

※ [(BTReader) Requires permission]

- android.Manifest.permission.BLUETOOTH

※ Optiamal RF configuration values.

	Value
RF Mode	1

Sesstion	**S1**
Toggle	**OFF**
Singulation	**10**

RF_PerformInventoryWithLocating		
Declare		**public int RF_PerformInventoryWithLocating(boolean turboMode, boolean enableSelection, boolean ignorePC)**
Description		Performs the inventory operation with locating
Parameter		**turboMode : 1(default)** - True : Continuous mode(Duty cycle = 0) - False : Non Continuous mode **enableSelection** - True : Select enable(Set RF_SetSelection API first.) - False : Select disable **ignorePC** - True : The tag data that removed PC field. - False : The tag data that included PC field.
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - INVENTORY = 5 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - INVENTORY = 5 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Mode Error** : SDConsts.RFResult.***MODE_ERROR*** = -6 **Battery Error** : SDConsts.RFResult.***LOW_BATTERY*** = -12

		Hotswap Error : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_PerformInventoryForLocating		
Declare		**public int RF_PerformInventoryForLocating(String epc)**
Description		Performs the inventory operation for locating
Parameter		**epc** - The epc value to locate
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - LOCATE = 17 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - LOCATE = 17 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Mode Error** : SDConsts.RFResult.***MODE_ERROR*** = -6 **Battery Error** : SDConsts.RFResult.***LOW_BATTERY*** = -12 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH** **※ Optiamal RF configuration values.**

	Value

RF Mode	**1**
Sesstion	**S0**
Toggle	**ON**
Singulation	**5**

RF_PerformInventoryWithPhaseFreq		
Declare		**public int RF_PerformInventoryWithPhaseFreq (boolean turboMode, boolean enableSelection, boolean ignorePC)**
Description		Performs the inventory operation with phase,frequency
Parameter		**turboMode : 1(default)** - True : Continuous mode(Duty cycle = 0) - False : Non Continuous mode **enableSelection** - True : Select enable(Set RF_SetSelection API first.) - False : Select disable **ignorePC** - True : The tag data that removed PC field. - False : The tag data that included PC field.
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - INVENTORY = 5 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - INVENTORY = 5 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Mode Error** : SDConsts.RFResult.***MODE_ERROR*** = -6

		Battery Error : SDConsts.RFResult.***LOW_BATTERY*** = -12 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of “RFResult” class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_PerformInventoryCustom		
Declare		**public int RF_PerformInventoryCustom (int RFMemType, int startlocation, int length, String accessPassword, boolean enableSelection)**
Description		Performs the inventory operation with other bank type
Parameter		**RFMemType:** - The memory bank type
(1 = EPC, 2 = TID, 3 = USER) **startlocation** - The first starting point(word base). 1word is 16bits **Length** - Read data length from startlocation(n = (16 * n) bits)(1 to 255) **accessPassword** - Access password for check (########) : Default 00000000 Import or set the password to set Tag's Access Permissions HEX Format : WORD(2-bytes) Length. **enableSelection** - Select enable(Set RF_SetSelection API first) False : Select disable
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - INVENTORY = 5 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of “RFResult” class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - INVENTORY = 5 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4

		Condition Error : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Mode Error** : SDConsts.RFResult.***MODE_ERROR*** = -6 **Battery Error** : SDConsts.RFResult.***LOW_BATTERY*** = -12 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 * **Can receive other error constant of “RFResult” class.**
Remark		※ **[(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_PerformInventoryWithRssiLimitation		
Declare		**public int RF_PerformInventoryWithRssiLimitation(boolean turboMode, boolean enableSelection, boolean ignorePC, int rssiLimitation)**
Description		Performs the inventory operation with RSSI Limitation
Parameter		**turboMode : 1(default)** - True : Continuous mode(Duty cycle = 0) - False : Non Continuous mode **enableSelection** - True : Select enable(Set RF_SetSelection API first.) - False : Select disable **ignorePC** - True : The tag data that removed PC field. - False : The tag data that included PC field. **rssiLimitation** - Set rssi limitation. If rssi limitation value is '-60', reader only received tag rssi's range from -30 to -60.(rssi range: -80 ~ -30). RSSI limitation setting disappears when 'RF_StopInventory()' is called.
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - INVENTORY = 5 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 * **Can receive other error constant of “RFResult” class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - INVENTORY = 5

		Enabled Error : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Mode Error** : SDConsts.RFResult.***MODE_ERROR*** = -6 **Battery Error** : SDConsts.RFResult.***LOW_BATTERY*** = -12 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_PerformInventoryEncoding		
Declare		**public int RF_PerformInventoryEncoding(boolean turboMode, boolean enableSelection, boolean ignorePC)**
Description		Performs the inventory operation with RSSI Limitation
Parameter		**turboMode : 1(default)** - True : Continuous mode(Duty cycle = 0) - False : Non Continuous mode **enableSelection** - True : Select enable(Set RF_SetSelection API first.) - False : Select disable **ignorePC** - True : The tag data that removed PC field. - False : The tag data that included PC field.
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - INVENTORY = 5 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - INVENTORY = 5

	Enabled Error : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Mode Error** : SDConsts.RFResult.***MODE_ERROR*** = -6 **Battery Error** : SDConsts.RFResult.***LOW_BATTERY*** = -12 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark	**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

	RF_SetEncodeInformation
Declare	**public int RF_SetEncodeInformation (int seqId, int targetEpcLength,** **String targetEpcdata, String accessPwdToWrite,** **int memoryBankTypeToWrite, int dataPosToWrite,** **int dataLenToWrite, String dataToWrite)**
Description	Writes to a specified bulk memory bank of tag during encoding Inventory (RF_PerformInventoryEncoding)
Parameter	**seqId** - seqeunc ID **targetEpcLength** - HEX form EPC length to mask **targetEpcdata** - HEX form EPC to mask HEX form of epc Data to be write **accessPassword** - **Access password for check (########) : Default 00000000** - Import or set the password to set Tag's Access Permissions HEX Format : WORD(2-bytes) Length **memoryBankTypeToWrite** - The memory bank type <BR>(0 = RESERVED, 1 = EPC, 2 = TID, 3 = USER) **dataPosToWrite** - The first starting point(word base). 1word is 16bits **dataLenToWrite** - HEX form of Data length to be write **dataToWrite** - HEX form of Data to be write

Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - WRITE_BULK = 60 **Access Password Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Other Error** SDConsts.RFResult.***OTHER_ERROR*** = -1 *** Can receive other error constant of “RFResult” class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - WRITE_BULK = 60 **Access Password Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Other Error** : SDConsts.RFResult.***OTHER_ERROR*** = -1 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of “RFResult” class.**
Remark		**This API may takes 10~10000 milliseconds. (depending on RF Access timeout value)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_StopInventoryEncoding		
Declare		**public int RF_StopInventoryEncoding ()**
Description		Stops the inventory operation
Parameter		void
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - STOP_INVENTORY = 5

		Other Error SDConsts.RFResult.***OTHER_ERROR*** = -1 **Inventory state Error** SDConsts.RFResult.***NOT_INVENTORY_STATE*** = -11 **Inventory stop Error** SDConsts.RFResult.***STOP_FAILED_TRY_AGAIN*** = -17 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - STOP_INVENTORY = 5 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Other Error** : SDConsts.RFResult.***OTHER_ERROR*** = -1 **Inventory state Error** SDConsts.RFResult.***NOT_INVENTORY_STATE*** = -11 **Inventory stop Error** SDConsts.RFResult.***STOP_FAILED_TRY_AGAIN*** = -17 *** Can receive other error constant of "RFResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		RF_StopInventory
Declare		**public int RF_StopInventory()**
Description		Stops the inventory operation
Parameter		void
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - STOP_INVENTORY = 5 **Other Error** SDConsts.RFResult.***OTHER_ERROR*** = -1 **Inventory state Error** SDConsts.RFResult.***NOT_INVENTORY_STATE*** = -11 **Inventory stop Error** SDConsts.RFResult.***STOP_FAILED_TRY_AGAIN*** = -17 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - STOP_INVENTORY = 5 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5

		Other Error : SDConsts.RFResult.***OTHER_ERROR*** = -1 **Inventory state Error** SDConsts.RFResult.***NOT_INVENTORY_STATE*** = -11 **Inventory stop Error** SDConsts.RFResult.***STOP_FAILED_TRY_AGAIN*** = -17 * **Can receive other error constant of “RFResult” class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		RF_BlockWrite
Declare		**public int RF_BlockWrite(int RFMemType, int offset, String data, String accessPassword)**
Description		Allows to writing multiple words in a Tag's Reserved, EPC, TID, or User memory using a single command
Parameter		**RFMemType (RESERVED(0) ~ USER(3))** - The memory bank type (0 = RESERVED, 1=EPC, 2=TID, 3=USER) **offset** - The offset, in the memory bank, of the first 16-bit word to write. **data** - UNICODE string to write. **accessPassword** - **Access password for check (########) : Default 00000000** - Import or set the password to set Tag's Access Permissions HEX Format : WORD(2-bytes) Length
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - BLOCK_WRITE = 12 **Memory Type Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Access Password Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 * **Can receive other error constant of “RFResult” class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - BLOCK_WRITE = 12

		Memory Type Error SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Access Password Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ Reference (3.2.RFMemType)** **This API may takes 10~10000 milliseconds. (depending on RF Access timeout value)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		RF_BlockPermalock
Declare		**public int RF_BlockPermalock** **(int blockPtr, int blockRange, int action, String accessPassword)**
Description		Allows to permalock multiple words in a Tag's Reserved, EPC, TID, or User memory with a single command, or read the permalock status of the memory blocks in a Tag's User memory
Parameter		**blockPtr** - Only 0 can be specified **blockRange** - Only 1 can be specified **action** - 0 : Retain current permalock setting - 1 : Assert permalock **accessPassword** - **Access password for check (########) : Default 00000000** - Import or set the password to set Tag's Access Permissions HEX Format : WORD(2-bytes) Length
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - BLOCK_PERMALOCK = 13 **Access Password Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4

		Connected Error SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - BLOCK_PERMALOCK = 13 **Access Password Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**This API may takes 10~10000 milliseconds. (depending on RF Access timeout value)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		RF_BlockErase
Declare		**public int RF_BlockErase(int RFMemType, int offset, int count, String accessPassword)**
Description		Erases tag
Parameter		**RFMemType** - The memory bank type (0 = RESERVED, 1=EPC, 2=TID, 3=USER) **offset** - The offset of the first 16-bit word, where zero is the first 16-bit word in the memory bank, to erase in the specified memory bank. **count** - The number of 16-bit words to be erased in the tag's specified memory bank. This parameter must contain a value between 1 and 255, inclusive. **accessPassword** - **Access password for check (########) : Default 00000000** - Import or set the password to set Tag's Access Permissions HEX Format : WORD(2-bytes) Length
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - BLOCK_ERASE = 14

		Memory Type Error SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Access Password Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - BLOCK_ERASE = 14 **Memory Type Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Access Password Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**This API may takes 10~10000 milliseconds. (depending on RF Access timeout value)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_KILL		
Declare		**public int RF_KILL(String killPassword, String accessPassword, boolean enableSelection)**
Description		Kills tag
Parameter		**killPassword** - kill password, HEX form **accessPassword** - **Access password for check (########) : Default 00000000** - Import or set the password to set Tag's Access Permissions HEX Format : WORD(2-bytes) Length **enableSelection** - True : Select enable(Set RF_SetSelection API first.) - False : Select disable
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0

		[Auto-update message from SLED] **SDConsts.RFCmdMsg** - KILL = 16 **Kill Password Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Access Password Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Other Error** SDConsts.RFResult.***OTHER_ERROR*** = -1 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - KILL = 16 **Kill Password Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Access Password Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Other Error** : SDConsts.RFResult.***OTHER_ERROR*** = -1 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**This API may takes 10~10000 milliseconds. (depending on RF Access timeout value)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_LOCK	
Declare	**public int RF_LOCK(String lockmask, String action, String accessPassword, boolean enableSelection)**
Description	Locks by accessing directly to memory of tag.
Parameter	Mask parameter can be added in order to lock the same tag as any mask. In this case, action, accessPassword can be set in set control packet. - Format <**lockmask**/**action** State>

9	8	7	6	**5**	**4**	**3**	**2**	**1**	**0**	**Bit Offset**
Kill Pwd		AccessPwd		EPC		TID		User		Memory Field
pwd	lock	pwd	lock	pwd	lock	pwd	lock	pwd	Lock	

<accessPassword/Lock State>

Pwd	Lock	Comments
0	0	Accessible
0	1	Accessible(No change)
1	0	Pwd Accessible
1	1	Not Accessible(No change)

- Only memory field that mask of lock command is equivalent to 1 is executed action, it can uses "11" or "00" for Action Mask. : (Don't use 10 or 01)
- Access item is read/write for kill PWD and accessPWD, and is write only for remaining memory field.
- Password to access tag set(Lock) password is set using set control command.

enableSelection

- True : Select enable(Set RF_SetSelection API first.)
- False : Select disable

Return **Reader**

Success Constants.RFResult.***SUCCESS*** = 0

[Auto-update message from SLED]

SDConsts.RFCmdMsg

- LOCK = 15

Lock mask Error SDConsts.RFResult.***ARGUMENT_ERROR*** = -3

Action Error SDConsts.RFResult.***ARGUMENT_ERROR*** = -3

Access Password Error SDConsts.RFResult.***ARGUMENT_ERROR*** = -3

Serial Error SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4

Condition Error SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7

Command State Error SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4

Connected Error SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5

Other Error SDConsts.RFResult.***OTHER_ERROR*** = -1

*** Can receive other error constant of "RFResult" class.**

BTReader

Success Constants.RFResult.***SUCCESS*** = 0

[Auto-update message from SLED]

SDConsts.RFCmdMsg

- LOCK = 15

Lock mask Error SDConsts.RFResult.***ARGUMENT_ERROR*** = -3

Action Error SDConsts.RFResult.***ARGUMENT_ERROR*** = -3

		Access Password Error SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Other Error** : SDConsts.RFResult.***OTHER_ERROR*** = -1 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ Reference (3.2.RFResult)** **This API may takes 10~10000 milliseconds. (depending on RF Access timeout value)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_READ

Declare		**public int RF_READ(int RFMemType, int startlocation, int length, String accessPassword, boolean enableSelection)**
Description		Reads a specified memory bank of tag
Parameter		**RFMemType** - The memory bank Type (0 = RESERVED, 1=EPC, 2=TID, 3=USER) **startlocation** - The first starting point(word base). 1word is 16bits. **length** - Read data length from startlocation(n = (16 * n) bits)(1 to 255) **accessPassword** - **Access password for check (########) : Default 00000000** - Import or set the password to set Tag's Access Permissions HEX Format : WORD(2-bytes) Length **enableSelection** - True : Select enable(Set RF_SetSelection API first.) - False : Select disable
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - READ = 7 **Memory Type Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Access Password Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3

		Serial Error SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Other Error** SDConsts.RFResult.***OTHER_ERROR*** = -1 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - READ = 7 **Memory Type Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Access Password Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Other Error** : SDConsts.RFResult.***OTHER_ERROR*** = -1 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ Reference (3.2.RFResult)** **This API may takes 10~10000 milliseconds. (depending on RF Access timeout value)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_WRITE	
Declare	**public int RF_WRITE(int RFMemType, int startlocation, String data,** **String accessPassword, boolean enableSelection)**
Description	Writes to a specified memory bank of tag
Parameter	**RFMemType** - The memory bank Type (0 = RESERVED, 1=EPC, 2=TID, 3=USER) **startlocation** - The first starting point(word base). 1word is 16bits. **data** - HEX form of Data to be write **accessPassword** - **Access password for check (########) : Default 00000000**

		- Import or set the password to set Tag's Access Permissions HEX Format : WORD(2-bytes) Length **enableSelection** - True : Select enable(Set RF_SetSelection API first.) - False : Select disable
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - WRITE = 8 **Memory Type Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Access Password Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Other Error** SDConsts.RFResult.***OTHER_ERROR*** = -1 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - WRITE = 8 **Memory Type Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Access Password Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Other Error** : SDConsts.RFResult.***OTHER_ERROR*** = -1 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ Reference (3.2.RFResult)** **This API may takes 10~10000 milliseconds. (depending on RF Access timeout value)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		RF_WriteAccessPassword
Declare		**public int RF_WriteAccessPassword(String data, String accessPassword, boolean enableSelection)**
Description		Writes to access password of a specific tag
Parameter		**data** - **Tag password (########) : Default 00000000** - Import or set the password to set Tag's Access Permissions HEX Format : WORD(2-bytes) Length **accessPassword** - **Access password for check (########) : Default 00000000** - Import or set the password to set Tag's Access Permissions HEX Format : WORD(2-bytes) Length **enableSelection** - True : Select enable(Set RF_SetSelection API first.) - False : Select disable
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - WRITE_ACCESS_PASSWORD = 9 **Access Password Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Other Error** SDConsts.RFResult.***OTHER_ERROR*** = -1 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - WRITE_ACCESS_PASSWORD = 9 **Access Password Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Other Error** : SDConsts.RFResult.***OTHER_ERROR*** = -1 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37

		*** Can receive other error constant of “RFResult” class.**
Remark		**※ Reference (3.2.RFResult)** **This API may takes 10~10000 milliseconds. (depending on RF Access timeout value)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_WriteKillPassword		
Declare		**public int RF_WriteKillPassword** **(String data, String accessPassword, boolean enableSelection)**
Description		Writes Kill password of a specific tag
Parameter		**data** - **Tag Kill password (########) : Default 00000000** - Import or set the password to set Tag's Access Permissions HEX Format : WORD(2-bytes) Length **accessPassword** - **Access password for check (########) : Default 00000000** - Import or set the password to set Tag's Access Permissions HEX Format : WORD(2-bytes) Length **enableSelection** - True : Select enable(Set RF_SetSelection API first.) - False : Select disable
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - WRITE_KILL_PASSWORD = 10 **Access Password Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Other Error** SDConsts.RFResult.***OTHER_ERROR*** = -1 *** Can receive other error constant of “RFResult” class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - WRITE_KILL_PASSWORD = 10 **Access Password Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3

		Enabled Error : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Other Error** : SDConsts.RFResult.***OTHER_ERROR*** = -1 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ Reference (3.2.RFResult)** **This API may takes 10~10000 milliseconds. (depending on RF Access timeout value)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		RF_WriteTagID
Declare		**public int RF_WriteTagID(int startlocation, String data, String accessPassword, boolean enableSelection)**
Description		Writes to TagID of a specific tag and adjusts the PC bits according to the length of the TagID
Parameter		**startlocation** - The first starting point(word base). 1word is 16bits. **data** - HEX form of Data to be write **accessPassword** - **Access password for check (########) : Default 00000000** - Import or set the password to set Tag's Access Permissions HEX Format : WORD(2-bytes) Length **enableSelection** - True : Select enable(Set RF_SetSelection API first.) - False : Select disable
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - WRITE_TAG_ID = 11 **Access Password Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5

		Other Error SDConsts.RFResult.***OTHER_ERROR*** = -1 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - WRITE_TAG_ID = 11 **Access Password Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Other Error** : SDConsts.RFResult.***OTHER_ERROR*** = -1 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**This API may takes 10~10000 milliseconds. (depending on RF Access timeout value)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_BulkWrite	
Declare	**public int RF_BulkWrite(int reservedStartPos, String reservedData,** **int epcMemStartPos, String epcMemData,** **int userMemStartPos, String userMemData,** **String accessPassword, int scMemType,** **int selectStartPos, int selectMaskLengthBit, String mask)**
Description	Writes to a specified bulk or simple memory bank of tag
Parameter	**reservedData** - HEX form of reserved Data to be write **epcMemStartPos** - The first starting point(word base). 1word is 16bits **epcMemData** - HEX form of epc Data to be write **userMemStartPos** - The first starting point(word base). 1word is 16bits **userMemData** - HEX form of user Data to be write **accessPassword** - **Access password for check (########) : Default 00000000** - Import or set the password to set Tag's Access Permissions HEX Format :

		WORD(2-bytes) Length **scMemType** - The memory bank type <BR>(0 = RESERVED, 1 = EPC, 3 = USER) **selectStartPos** - The Value is base on number of characters **selectMaskLengthBit** - Length of the selected mask(bit) (ex. "3000" is 16bits, 30 - 1byte, 00- 1byte.) **Mask** - HEX form(ex. "3000", "1234ABFF"), Mask value is multiply of 2
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - WRITE_BULK = 60 **Access Password Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Other Error** SDConsts.RFResult.***OTHER_ERROR*** = -1 *** Can receive other error constant of “RFResult” class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - WRITE_BULK = 60 **Access Password Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Other Error** : SDConsts.RFResult.***OTHER_ERROR*** = -1 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of “RFResult” class.**
Remark		**This API may takes 10~10000 milliseconds. (depending on RF Access timeout value)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		RF_WriteSwitchMode
Declare		**public int RF_WriteSwitchMode(int switchMode, String newAccessPassword,** **String oldAccessPassword,** **int scMemType, int selectStartPos,** **int selectMaskLengthBit, String mask)**
Description		Writes to access password of a specific tag with private or narmal mode
Parameter		**SwitchMode** - select mode(private:0 or normal:1) **newAccessPassword** - To change Tag password**(########) : Default 00000000** - Import or set the password to set Tag's Access Permissions HEX Format : WORD(2-bytes) Length **OldAccessPassword** - **Access password for check (########) : Default 00000000** - Import or set the password to set Tag's Access Permissions HEX Format : WORD(2-bytes) Length **ScMemType** - The memory bank type <BR>(0 = RESERVED, 1 = EPC, 3 = USER) **SelectStartPos** - The Value is base on number of characters **SelectMaskLengthBit** - Length of the selected mask(bit) (ex. "3000" is 16bits, 30 - 1byte, 00- 1byte) **Mask** - HEX form(ex. "3000", "1234ABFF"), Mask value is multiply of 2
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.RFCmdMsg** - WRITE_PRIVATE_MODE = 61 - WRITE_NORMAL_MODE = 62 **Access Password Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Other Error** SDConsts.RFResult.***OTHER_ERROR*** = -1 **Not Supported Error** SDConsts.SDCommonResult.***NOT_SUPPORTED_API*** = -36 *** Can receive other error constant of “RFResult” class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 [Auto-update message from SLED]

		SDConsts.RFCmdMsg - WRITE_PRIVATE_MODE = 61 - WRITE_NORMAL_MODE = 62 **Access Password Error** SDConsts.RFResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Other Error** : SDConsts.RFResult.***OTHER_ERROR*** = -1 **Not Supported Error** SDConsts.SDCommonResult.***NOT_SUPPORTED_API*** = -36 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**This API may takes 10~10000 milliseconds. (depending on RF Access timeout value)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_SetEnableChannels		
Declare		**public int RF_SetEnableChannels(int region, String[] channels)**
Description		Sets channel in region
Parameter		**region** - Target region **channels** - Target channel table
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Other Error** SDConsts.RFResult.***OTHER_ERROR*** = -1 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4

		Other Error : SDConsts.RFResult.***OTHER_ERROR*** = -1 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**In case of this API, Run time during about 0 ~ 8 seconds is required. It sends related callback message(REGION_CHANGE_START(21) -> REGION_CHANGE_END(22)) at the beginning and the end.)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_SetEnableChannels		
Declare		**public int RF_SetEnableChannels(String isoCode, String[] channels)**
Description		Sets channel in region
Parameter		**isoCode** - Target region's iso-code **channels** - Target channel table
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Other Error** : SDConsts.RFResult.***OTHER_ERROR*** = -1 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**In case of this API, Run time during about 0 ~ 8 seconds is required. It sends related callback message(REGION_CHANGE_START(21) -> REGION_CHANGE_END(22)) at the beginning and the end.)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_SetRegionISO		
Declare		**public int RF_SetRegionISO(String iso_code)**
Description		Sets the Region value of RFID radio module
Parameter		**Iso_code** - ISO 3166-1 alfa-3 / ISO 3166-1 alfa-2, refer Class SDConsts.RFISORegion
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 **Range Error** : SDConsts.RFResult.ARGUMENT_ERROR = -3 **Serial Error** : SDConsts.RFResult.OTHER_CMD_RUNNING_ERROR = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Region Error** : SDConsts.RFResult.OTHER_ERROR = -1 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 **Range Error** : SDConsts.RFResult.ARGUMENT_ERROR = -3 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Other Error** : SDConsts.RFResult.***OTHER_ERROR*** = -1 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**In case of this API, Run time during about 0 ~ 8 seconds is required. It sends related callback message(REGION_CHANGE_START(21) -> REGION_CHANGE_END(22)) at the beginning and the end.)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_checkRegionISO		
Declare		**public boolean RF_checkRegionISO(String iso_code)**
Description		Gets result that compare current Region and param(iso code)
Parameter		**Iso_code** - ISO 3166-1 alfa-3 / ISO 3166-1 alfa-2, refer Class SDConsts.RFISORegion

Return	**Reader**	**Success** true(return true If the iso code is same as the current region) **Fail :** False
	BTReader	**Success** true(return true If the iso code is same as the current region) **Fail :** False
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		RF_GetEnableChannels
Declare		**public String[] RF_GetEnableChannels()**
Description		Gets enabled channels
Parameter		
Return	**Reader**	**Success** enabled channel table **Fail** null
	BTReader	**Success** enabled channel table **Fail** null
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		RF_GetDefaultChannels
Declare		**public String[] RF_GetDefaultChannels()**
Description		Gets selected region default channel table
Parameter		
Return	**Reader**	**Success** selected region default channel table **Fail** null
	BTReader	**Success** selected region default channel table **Fail** null
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		RF_SetLBTVaule
Declare		**public int RF_SetLBTVaule (int val)**
Description		Sets the LBT mode value of the RFID radio module(only use JP1/JP2)
Parameter		LBT OFF : 0 ,LBT ON : 1(default), LBT SCAN MODE : 3
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0

		Range Error : SDConsts.RFResult.ARGUMENT_ERROR = -3 **Serial Error** : SDConsts.RFResult.OTHER_CMD_RUNNING_ERROR = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Region Error** : SDConsts.RFResult.OTHER_ERROR = -1 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 **Range Error** : SDConsts.RFResult.ARGUMENT_ERROR = -3 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Other Error** : SDConsts.RFResult.***OTHER_ERROR*** = -1 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_GetLBTVaule		
Declare		**public int RF_GetLBTVaule()**
Description		Gets the LBT mode value of the RFID radio module(only use JP1/JP2)
Parameter		
Return	**Reader**	**Success** Value of the LBT State(LBT OFF : 0 ,LBT ON : 1(default), LBT SCAN MODE : 3) **Range Error** : SDConsts.RFResult.ARGUMENT_ERROR = -3 **Serial Error** : SDConsts.RFResult.OTHER_CMD_RUNNING_ERROR = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Region Error** : SDConsts.RFResult.OTHER_ERROR = -1 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Value of the LBT State(LBT OFF : 0 ,LBT ON : 1(default), LBT SCAN MODE : 3) **Range Error** : SDConsts.RFResult.ARGUMENT_ERROR = -3 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15

		Connected Error : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Other Error** : SDConsts.RFResult.***OTHER_ERROR*** = -1 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		RF_SetDYNRFMode
Declare		**public int RF_SetDYNRFMode(int val)**
Description		Set Dynamic mode behavior
Parameter		**val** This register has a single field which can contain 3 different values. Only the first two can be set by the host, the final value is only set by the firmware itself to indicate operating condition. Those values are listed below: - 0x0000 – Static Mode: Operate in a single Static RF Mode. This is the same behavior as previous versions of IndyMAC. - 0x0001 – Init Dynamic Mode: Initialize the Dynamic RF Mode. This allows the user to configure the Dynamic RF Mode without starting inventory. - 0x1000 – Run Dynamic Mode: Indicates that the Dynamic RF Mode is currently being run. This value is only set by the firmware itself.
Return	**Reader**	**Success** Constants.RFResult.SUCCESS = 0 **Range Error** : SDConsts.RFResult.ARGUMENT_ERROR = -3 **Serial Error** : SDConsts.RFResult.OTHER_CMD_RUNNING_ERROR = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Region Error** : SDConsts.RFResult.OTHER_ERROR = -1 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.SUCCESS = 0 **Range Error** : SDConsts.RFResult.ARGUMENT_ERROR = -3 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7

		Command State Error : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Other Error** : SDConsts.RFResult.***OTHER_ERROR*** = -1 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_GetDYNRFMode

Declare		**public int RF_GetDYNRFMode()**
Description		Get Dynamic mode behavior
Parameter		
Return	**Reader**	**Success** 1(Dynamic mdoe Enable) or 0(Dynamic mdoe Disable) **Serial Error** : SDConsts.RFResult.OTHER_CMD_RUNNING_ERROR = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** 1(Dynamic mdoe Enable) or 0(Dynamic mdoe Disable) **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_SetDYNStartQ

Declare		**public int RF_SetDYNStartQ(int qVal)**
Description		Sets Starting Q value in each of the dynamic RF Modes.
Parameter		**qVal** - Starting Q value in each of the dynamic RF Modes.
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0

		Serial Error : SDConsts.RFResult.OTHER_CMD_RUNNING_ERROR = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		RF_GetDYNStartQ
Declare		**public int RF_GetDYNStartQ()**
Description		Gets Starting Q value in each of the dynamic RF Modes.
Parameter		
Return	**Reader**	**Success** Value of the Starting Q value in each of the dynamic RF Modes. - The starting Q value to use. Valid values are 0-15, inclusive. starting Q Value must be greater than or equal to minimumQValue and less than or equal to maximumQValue. **Serial Error** : SDConsts.RFResult.OTHER_CMD_RUNNING_ERROR = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Value of the Starting Q value in each of the dynamic RF Modes. **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4

		Hotswap Error : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_SetDYNModeSquence		
Declare		**public int RF_SetDYNModeSquence(int sVal)**
Description		Defines the sequence of RF Modes that are used in the dynamic RF Mode sequence value.
Parameter		**sVal** - Starting RF Mode sequence value in each of the dynamic RF Modes.
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 **Serial Error** : SDConsts.RFResult.OTHER_CMD_RUNNING_ERROR = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_GetDYNModeSquence		
Declare		**public int RF_GetDYNModeSquence()**
Description		Gets the sequence of RF Modes that are used in the dynamic RF Mode sequence value.
Parameter		
Return	**Reader**	**Success** Starting Q value in each of the dynamic RF Modes. **Serial Error** : SDConsts.RFResult.OTHER_CMD_RUNNING_ERROR = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7

		Command State Error SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Value of the Starting Q value in each of the dynamic RF Modes. **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_SetDYNModeMinMaxMode		
Declare		**public int RF_SetDYNModeMinMaxMode(int mVal)**
Description		Sets the minimum/maximum Q value for each of the RF Modes in the sequence.
Parameter		**mVal** - Starting RF Mode sequence minimum/maximum Q value in each of the dynamic RF Modes.
Return	**Reader**	**Success** Constants.RFResult.***SUCCESS*** = 0 **Serial Error** : SDConsts.RFResult.OTHER_CMD_RUNNING_ERROR = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.RFResult.***SUCCESS*** = 0 **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_GetDYNModeMinMaxMode		
Declare		**public int RF_GetDYNModeMinMaxMode()**
Description		Gets the minimum/maximum Q value for each of the RF Modes in the sequence.
Parameter		
Return	**Reader**	**Success** the minimum/maximum Q value for each of the RF Modes in the sequence. **Serial Error** : SDConsts.RFResult.OTHER_CMD_RUNNING_ERROR = -4 **Condition Error** SDConsts.RFResult.*READER_OR_SERIAL_STATUS_ERROR* = -7 **Command State Error** SDConsts.RFResult.*OTHER_CMD_RUNNING_ERROR* = -4 **Connected Error** SDConsts.RFResult.*SD_NOT_CONNECTED* = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** the minimum/maximum Q value for each of the RF Modes in the sequence. **Enabled Error** : SDConsts.RFResult.*BLUETOOTH_NOT_ENABLED* = -15 **Connected Error** : SDConsts.RFResult.*SD_NOT_CONNECTED* = -5 **Block State Error** : SDConsts.RFResult.*OTHER_CMD_RUNNING_ERROR* = -4 **Condition Error** : SDConsts.RFResult.*READER_OR_SERIAL_STATUS_ERROR* = -7 **Command State Error** : SDConsts.RFResult.*OTHER_CMD_RUNNING_ERROR* = -4 **Hotswap Error** : SDConsts.RFResult.*ERROR_HOTSWAP_STATE* = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

RF_UpdateDYNProfile		
Declare		**public int RF_UpdateDYNProfile(String filepath)**
Description		Updates the Dynamic Profile.
Parameter		**filepath** - File path for Dynamic Profile text file update
Return	**Reader**	**Success** Constants.SDResult.SUCCESS = 0. **File path Error** : SDConsts.SDResult.ARGUMENT_ERROR = -3 **Serial Error** : SDConsts.RFResult.OTHER_CMD_RUNNING_ERROR = -4 **Condition Error** SDConsts.RFResult.*READER_OR_SERIAL_STATUS_ERROR* = -7 **Command State Error** SDConsts.RFResult.*OTHER_CMD_RUNNING_ERROR* = -4 **Connected Error** SDConsts.RFResult.*SD_NOT_CONNECTED* = -5

		*** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.SDResult.SUCCESS = 0. **File path Error** : SDConsts.SDResult.ARGUMENT_ERROR = -3 **Enabled Error** : SDConsts.RFResult.BLUETOOTH_NOT_ENABLED = -15 **Connected Error** : SDConsts.RFResult.SD_NOT_CONNECTED = -5 **Block State Error** : SDConsts.RFResult.OTHER_CMD_RUNNING_ERROR = -4 **Condition Error** : SDConsts.RFResult.READER_OR_SERIAL_STATUS_ERROR = -7 **Command State Error** : SDConsts.RFResult.OTHER_CMD_RUNNING_ERROR = -4 **Hotswap Error** : SDConsts.RFResult.ERROR_HOTSWAP_STATE = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ [(BTReader) Requires permission]** **android.Manifest.permission.BLUETOOTH** **※ Reference** **In case of this API, Run time during about 90 seconds is required. It sends related callback message(UPDATE_RF_DYN_MAC_START(53) -> UPDATE_RF_DYN_MAC_FW(54) -> UPDATE_RF_DYN_MAC_END(55)) at the beginning and the end.**

		RF_UpdateDYNProfileFCC
Declare		**public int RF_UpdateDYNProfileFCC()**
Description		Updates the Dynamic Profile for FCC device.
Parameter		
Return	**Reader**	**Success** Constants.SDResult.SUCCESS = 0. **Serial Error** : SDConsts.RFResult.OTHER_CMD_RUNNING_ERROR = -4 **Condition Error** SDConsts.RFResult.READER_OR_SERIAL_STATUS_ERROR = -7 **Command State Error** SDConsts.RFResult.OTHER_CMD_RUNNING_ERROR = -4 **Connected Error** SDConsts.RFResult.SD_NOT_CONNECTED = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.SDResult.SUCCESS = 0. **Enabled Error** : SDConsts.RFResult.BLUETOOTH_NOT_ENABLED = -15 **Connected Error** : SDConsts.RFResult.SD_NOT_CONNECTED = -5 **Block State Error** : SDConsts.RFResult.OTHER_CMD_RUNNING_ERROR = -4 **Condition Error** : SDConsts.RFResult.READER_OR_SERIAL_STATUS_ERROR = -7 **Command State Error** : SDConsts.RFResult.OTHER_CMD_RUNNING_ERROR = -4 **Hotswap Error** : SDConsts.RFResult.ERROR_HOTSWAP_STATE = -37 *** Can receive other error constant of "RFResult" class.**

Remark		※ **[(BTReader) Requires permission]** **android.Manifest.permission.BLUETOOTH**
		※ **Reference** **In case of this API, Run time during about 90 seconds is required. It sends related callback message(UPDATE_RF_DYN_MAC_START(53) -> UPDATE_RF_DYN_MAC_FW(54) -> UPDATE_RF_DYN_MAC_END(55)) at the beginning and the end.**

		RF_UpdateDYNProfileEU
Declare		**public int RF_UpdateDYNProfileEU()**
Description		Updates the Dynamic Profile for EU device.
Parameter		
Return	**Reader**	**Success** Constants.SDResult.SUCCESS = 0. **Serial Error** : SDConsts.RFResult.OTHER_CMD_RUNNING_ERROR = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.SDResult.SUCCESS = 0. **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		※ **[(BTReader) Requires permission]** **android.Manifest.permission.BLUETOOTH**
		※ **Reference** **In case of this API, Run time during about 90 seconds is required. It sends related callback message(UPDATE_RF_DYN_MAC_START(53) -> UPDATE_RF_DYN_MAC_FW(54) -> UPDATE_RF_DYN_MAC_END(55)) at the beginning and the end.**

		RF_StartCarrierWave
Declare		**public int RF_StartCarrierWave(int freq)**
Description		Start one channel carrier wave
Parameter		**freq** Channel value
Return	**Reader**	**Success** Constants.SDResult.SUCCESS = 0. **Serial Error** : SDConsts.RFResult.OTHER_CMD_RUNNING_ERROR = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.SDResult.SUCCESS = 0. **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ [(BTReader) Requires permission]** **android.Manifest.permission.BLUETOOTH**
		.

		RF_StopCarrierWave
Declare		**public int RF_StopCarrierWave()**
Description		Stop one channel carrier wave
Parameter		
Return	**Reader**	**Success** Constants.SDResult.SUCCESS = 0. **Serial Error** : SDConsts.RFResult.OTHER_CMD_RUNNING_ERROR = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.SDResult.SUCCESS = 0.

		Enabled Error : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ [(BTReader) Requires permission]** **android.Manifest.permission.BLUETOOTH**
		.

RF_SetRFIDProtocolType

Declare		**public int RF_SetRFIDProtocolType(int Type)**
Description		Enable/disable Gen2/Gen2X/GEN2_GEN2X protocol
Parameter		**Type** Default 0(Gen2),1(Gen2x), 2(GEN2 and GEN2X)
Return	**Reader**	**Success** Constants.SDResult.SUCCESS = 0. **Serial Error** : SDConsts.RFResult.OTHER_CMD_RUNNING_ERROR = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.SDResult.SUCCESS = 0. **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ [(BTReader) Requires permission]** **android.Manifest.permission.BLUETOOTH**
		.

		RF_GetRFIDProtocolType
Declare		**public int RF_SetRFIDProtocolType(int Type)**
Description		Get enabled protocol's type
Parameter		
Return	**Reader**	**Success** Protocol's type **Serial Error** : SDConsts.RFResult.OTHER_CMD_RUNNING_ERROR = -4 **Condition Error** SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "RFResult" class.**
	BTReader	**Success** Constants.SDResult.SUCCESS = 0. **Enabled Error** : SDConsts.RFResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.RFResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.RFResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.RFResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.RFResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "RFResult" class.**
Remark		**※ [(BTReader) Requires permission]** **android.Manifest.permission.BLUETOOTH**
		.

■ SD APIs

		SD_Open
Declare		**public boolean SD_Open()**
Description		Open SLED, Ready for all communication(Serial, Barcode and so on)
Parameter		void
Return	**Reader**	**Success :** True **Fail :** False
	BTReader	**Success :** True **Fail** : False
Remark		**- SD_Open API has function of barcode open** **- Both RFR900 and RFR901 are available, but RFR901 must have the following or higher versions of image for each OS.** **>> [A9/A10]All, [A7]20180218~, [A6]20180130~, [A5]20180129~**

SD_Open		
Declare		**public boolean SD_Open(String clientId)**
Description		Open SLED, Ready for all communication with specific client feature. (Serial, Barcode and so on)
Parameter		clientId - Company id String
Return	**Reader**	**Success :** True **Fail :** False
	BTReader	**Success :** True **Fail** : False
Remark		**SD_Open API has function of barcode open**

SD_Close		
Declare		**public boolean SD_Close()**
Description		Close SLED, Close all opened communication(Serial, Barcode and so on)
Parameter		void
Return	**Reader**	**Success :** True **Fail :** False
	BTReader	**Success :** True **Fail** : False
Remark		**SD_Close API has function of barcode close** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH** - **android.Manifest.permission.BLUETOOTH_ADMIN**

SD_GetVersion		
Declare		**public String SD_GetVersion()**
Description		Gets the firmware version of SLED
Parameter		void
Return	**Reader**	**Success :** Version of the SLED firmware **Serial Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** = “Error”

		*** Can receive other error constant of “SDResult” class.**
	BTReader	**Success :** Version of the SLED firmware **Enabled Error** : SDConsts.SDResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : “Error” **Block State Error** : SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.SDResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of “SDResult” class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SD_GetBootLoaderVersion		
Declare		**public String SD_GetBootLoaderVersion()**
Description		Gets the boot loader version of SLED
Parameter		void
Return	**Reader**	**Success :** Version of the SLED boot loader **Serial Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** = “Error” **Not Supported Error** SDConsts.SDResult.***NOT_SUPPORTED_API*** = “Not Supported API" *** Can receive other error constant of “SDResult” class.**
	BTReader	**Success :** Version of the SLED boot loader **Enabled Error** : SDConsts.SDResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : “Error” **Block State Error** : SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -44 **Hotswap Error** : SDConsts.SDResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of “SDResult” class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SD_GetBatteryStatus		
Declare		**public int SD_GetBatteryStatus()**
Description		Gets the battery status(value) of the SLED
Parameter		void
Return	**Reader**	**Success :** Value of the Battery status **(MIN(0) ~ MAX(100))** **Serial Error** SDConsts.SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.SDBatteryState.***READER_OR_SERIAL_STATUS_ERROR***= -7 **Command State Error** SDConsts.SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.SDBatteryState.***SD_NOT_CONNECTED***= -5 *** Can receive other error constant of "SDBatteryState" class.**
	BTReader	**Success :** Value of the Battery status **(MIN(0) ~ MAX(100))** **Enabled Error** : SDConsts.SDBatteryState.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SDBatteryState.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SDBatteryState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -44 **Hotswap Error** : SDConsts.SDBatteryState.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SDBatteryState" class.**
Remark		**※ Reference (3.2.SDBatteryState)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SD_GetTriggerMode		
Declare		**public int SD_GetTriggerMode()**
Description		Gets the trigger mode of the SLED
Parameter		void
Return	**Reader**	**Success :** Value of the trigger mode **(RFID(0) , BARCODE(1))** **Serial Error** SDConsts.SDTriggerMode.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.SDTriggerMode.***READER_OR_SERIAL_STATUS_ERROR***= -7 **Command State Error** SDConsts.SDTriggerMode.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.SDTriggerMode.***SD_NOT_CONNECTED***= -5 *** Can receive other error constant of "SDTriggerMode" class.**
	BTReader	**Success :** Value of the trigger mode **(RFID(0) , BARCODE(1))** **Enabled Error** : SDConsts.SDTriggerMode.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SDTriggerMode.***SD_NOT_CONNECTED*** = -5

		Block State Error : SDConsts.SDTriggerMode.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SDTriggerMode.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SDTriggerMode.***OTHER_CMD_RUNNING_ERROR*** = -44 **Hotswap Error** : SDConsts.SDTriggerMode.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SDTriggerMode" class.**
Remark		**※ Reference (3.2.SDTriggerMode, 3.5 Barcode mode)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

	SD_SetTriggerMode	
Declare		**public int SD_SetTriggerMode(int SDTriggerMode)**
Description		Sets the trigger mode of SLED
Parameter		**SDTriggerMode** - Trigger mode **(0 : RFID / 1 : Barcode)**
Return	**Reader**	**Success** Constants.SDResult.***SUCCESS*** = 0 **Range Error** SDConsts.SDResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.SDResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "SDResult" class.**
	BTReader	**Success** Constants.SDResult.***SUCCESS*** = 0 **Range Error** : SDConsts.SDResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** : SDConsts.SDResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SDResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -44 **Hotswap Error** : SDConsts.SDResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SDResult" class.**
Remark		**※ Reference (3.5) Barcode mode** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SD_Connect	
Declare	**public int SD_Connect()**
Description	Connects to SLED(SLED DEV START)
Parameter	void
Reader Return	**Success** Constants.SDResult.***SUCCESS*** = 0 **Serial Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **SD Connect Error** SDConsts.SDResult.***DUP_CMD_ERROR*** = -8 **Command State Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.SDResult.***ALREADY_CONNECTED*** = -10 **Timeout Error** SDConsts.SDResult.***ACCESS_TIMEOUT*** = -32 *** Can receive other error constant of "SDResult" class.**
Remark	**Operate it after receiving wakeup callback message*(SLED_WAKEUP(47))*** **※ This API is only for Serial interface(Reader)**

SD_Disconnect	
Declare	**public int SD_Disconnect()**
Description	Disconnects to SLED(SLED DEV STOP)
Parameter	void
Reader Return	**Success** Constants.SDResult.***SUCCESS*** = 0 **Condition Error** SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **SD Connect Error** SDConsts.SDResult.***DUP_CMD_ERROR*** = -8 **Connected Error** SDConsts.SDResult.***ALREADY_DISCONNECTED*** = -9 **Timeout Error** SDConsts.SDResult.***ACCESS_TIMEOUT*** = -32 *** Can receive other error constant of "SDResult" class.**
Remark	**※ This API is only for Serial interface(Reader)**

SD_SetBuzzerLevel	
Declare	**public int SD_SetBuzzerLevel(int SDBuzzerLevel)**
Description	Sets the buzzer level of the SLED
Parameter	**SDBuzzerLevel :** Argument is Buzzer Level - HIGH = 2 - MID = 1 - LOW = 0

Return	**Reader**	**Success** Constants.SDResult.***SUCCESS*** = 0 **Range Error** SDConsts.SDResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.SDResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "SDResult" class.**
	BTReader	**Success** Constants.SDResult.***SUCCESS*** = 0 **Range Error** : SDConsts.SDResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** : SDConsts.SDResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SDResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -44 **Hotswap Error** : SDConsts.SDResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SDResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SD_GetBuzzerLevel

Declare		**public int SD_GetBuzzerLevel()**
Description		Gets the buzzer level of the SLED
Parameter		void
Return	**Reader**	**Success :** Value of the buzzer level **(LOW(0) ~ HIGH(2))** **Serial Error** SDConsts.SDBuzzerLevel.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.SDBuzzerLevel.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.SDBuzzerLevel.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.SDBuzzerLevel.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "SDBuzzerLevel" class.**
	BTReader	**Success :** Value of the buzzer level **(LOW(0) ~ HIGH(2))** **Enabled Error** : SDConsts.SDBuzzerLevel.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SDBuzzerLevel.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SDBuzzerLevel.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SDBuzzerLevel.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SDBuzzerLevel.***OTHER_CMD_RUNNING_ERROR*** = -44

		Hotswap Error : SDConsts.SDBuzzerLevel.*ERROR_HOTSWAP_STATE* = -37 *** Can receive other error constant of "SDBuzzerLevel" class.**
Remark		**※ Reference (3.2.SDBuzzerLevel)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SD_SetAutoSleepTimeout		
Declare		**public int SD_SetAutoSleepTimeout(int SDSleepTimeout)**
Description		Sets the auto-sleep timeout of the SLED
Parameter		**SDSleepTimeout** : Timeout argument(0~6) - NO_SLEEP = 0 - (MINIMUM) SEC_15 = 1 - (MAXIMUM) MIN_10 = 6
Return	**Reader**	**Success** Constants.SDResult.*SUCCESS* = 0 **Range Error** SDConsts.SDResult.*ARGUMENT_ERROR* = -3 **Serial Error** SDConsts.SDResult.*OTHER_CMD_RUNNING_ERROR* = -4 **Condition Error** SDConsts.SDResult.*READER_OR_SERIAL_STATUS_ERROR* = -7 **Command State Error** SDConsts.SDResult.*OTHER_CMD_RUNNING_ERROR* = -4 **Connected Error** SDConsts.SDResult.*SD_NOT_CONNECTED* = -5 *** Can receive other error constant of "SDResult" class.**
	BTReader	**Success** Constants.SDResult.*SUCCESS* = 0 **Range Error** : SDConsts.SDResult.*ARGUMENT_ERROR* = -3 **Enabled Error** : SDConsts.SDResult.*BLUETOOTH_NOT_ENABLED* = -15 **Connected Error** : SDConsts.SDResult.*SD_NOT_CONNECTED* = -5 **Block State Error** : SDConsts.SDResult.*OTHER_CMD_RUNNING_ERROR* = -4 **Condition Error** : SDConsts.SDResult.*READER_OR_SERIAL_STATUS_ERROR* = -7 **Command State Error** : SDConsts.SDResult.*OTHER_CMD_RUNNING_ERROR* = -4 **Hotswap Error** : SDConsts.SDResult.*ERROR_HOTSWAP_STATE* = -37 *** Can receive other error constant of "SDResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SD_GetAutoSleepTimeout		
Declare		**public int SD_GetAutoSleepTimeout()**
Description		Gets the auto-sleep timeout of the SLED

Parameter		Void
Return	**Reader**	**Success :** Value of the auto-sleep timeout **(NO_SLEEP(0) ~ MIN_10(6))** **Serial Error** SDConsts.SDSleepTimeout.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.SDSleepTimeout.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.SDSleepTimeout.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.SDSleepTimeout.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "SDSleepTimeout" class.**
	BTReader	**Success :** Value of the auto-sleep timeout **(NO_SLEEP(0) ~ MIN_10(6))** **Enabled Error** : SDConsts.SDSleepTimeout.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SDSleepTimeout.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SDSleepTimeout.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SDSleepTimeout.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SDSleepTimeout.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.SDSleepTimeout.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SDSleepTimeout" class.**
Remark		**※ Reference (3.2.SDSleepTimeout)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

	SD_GetConnectState
Declare	**public int SD_GetConnectState()**
Description	Gets the connection state with the SLED
Parameter	void
Reader Return	**Success :** Value of the connect state **(DISCONNECTED(0), CONNECTED(1))** **Serial Error** SDConsts.SDConnectState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.SDConnectState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.SDConnectState.***OTHER_CMD_RUNNING_ERROR*** = -4 *** Can receive other error constant of "SDConnectState" class.**
Remark	**※ Reference (3.2.SDConnectState)** **※ This API is only for Serial interface(Reader)**

	SD_SetBuzzerEnable
Declare	**public int SD_SetBuzzerEnable(int SDBuzzerMute)**

Description		Sets the buzzer enable state of the SLED
Parameter		**SDBuzzerMute** - On : 1 - Off : 0
Return	**Reader**	**Success** Constants.SDResult.***SUCCESS*** = 0 **Range Error** SDConsts.SDResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.SDResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "SDResult" class.**
	BTReader	**Success** Constants.SDResult.***SUCCESS*** = 0 **Range Error** : SDConsts.SDResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** : SDConsts.SDResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SDResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.SDResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SDResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SD_GetBuzzerState

Declare		**public int SD_SetBuzzerEnable()**
Description		Gets the buzzer enable state of the SLED
Parameter		void
Return	**Reader**	**Success :** Value of the buzzer state **(MUTE(0), NOISY(1))** **Serial Error** SDConsts.SDBuzzerState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.SDBuzzerState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.SDBuzzerState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.SDBuzzerState.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "SDBuzzerState" class.**
	BTReader	**Success :** Value of the buzzer state **(MUTE(0), NOISY(1))** **Enabled Error** : SDConsts.SDBuzzerState.***BLUETOOTH_NOT_ENABLED*** = -15

		Connected Error : SDConsts.SDBuzzerState.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SDBuzzerState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SDBuzzerState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SDBuzzerState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.SDBuzzerState.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of “SDBuzzerState” class**
Remark		**※ Reference (3.2.SDBuzzerState)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SD_SetTagBuzzerEnable

Declare		**public int SD_SetTagBuzzerEnable(int SDTagBuzzerState)**
Description		Sets the inventory buzzer enable state of the SLED
Parameter		**SDTagBuzzerState** - On : 1 - Off : 0
Return	**Reader**	**Success** Constants.SDResult.***SUCCESS*** = 0 **Range Error** SDConsts.SDResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.SDResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of “SDResult” class.**
	BTReader	**Success** Constants.SDResult.***SUCCESS*** = 0 **Range Error** : SDConsts.SDResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** : SDConsts.SDResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SDResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.SDResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of “SDResult” class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SD_GetTagBuzzerState		
Declare		**public int SD_GetTagBuzzerState()**
Description		Gets the inventory buzzer enable state of the SLED
Parameter		void
Return	**Reader**	**Success :** Value of the buzzer state **(Off(0), On(1))** **Serial Error** SDConsts.SDBuzzerState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.SDBuzzerState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.SDBuzzerState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.SDBuzzerState.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "SDBuzzerState" class.**
	BTReader	**Success :** Value of the buzzer state **(Off(0), On(1))** **Enabled Error** : SDConsts.SDBuzzerState.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SDBuzzerState.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SDBuzzerState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SDBuzzerState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SDBuzzerState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.SDBuzzerState.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SDBuzzerState" class**
Remark		**※ Reference (3.2.SDTagBuzzerState)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SD_SetTagBuzzerSound		
Declare		**public int SD_SetTagBuzzerSound()**
Description		Make buzzer sound of the SLED
Parameter		void
Return	**Reader**	**Success** Constants.SDResult.***SUCCESS*** = 0 **Serial Error** SDConsts.SDBuzzerState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.SDBuzzerState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.SDBuzzerState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.SDBuzzerState.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "SDBuzzerState" class.**
	BTReader	**Success** Constants.SDResult.***SUCCESS*** = 0 **Enabled Error** : SDConsts.SDBuzzerState.***BLUETOOTH_NOT_ENABLED*** = -15

		Connected Error : SDConsts.SDBuzzerState.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SDBuzzerState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SDBuzzerState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SDBuzzerState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.SDBuzzerState.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SDBuzzerState" class**
Remark		**※ Reference (3.2.SDTagBuzzerState)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SD_SetLEDEnable		
Declare		**public int SD_SetLEDEnable(int SDLEDState)**
Description		Sets the LED enable state of the SLED
Parameter		**SDLEDState** - Enable : 1 - Disable : 0
Return	**Reader**	**Success** Constants.SDResult.***SUCCESS*** = 0 **Range Error** SDConsts.SDResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.SDBuzzerState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.SDBuzzerState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.SDBuzzerState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.SDBuzzerState.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "SDBuzzerState" class.**
	BTReader	**Success** Constants.SDResult.***SUCCESS*** = 0 **Enabled Error** : SDConsts.SDBuzzerState.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SDBuzzerState.***SD_NOT_CONNECTED*** = -5 **Range Error** : SDConsts.SDResult.***ARGUMENT_ERROR*** = -3 **Block State Error** : SDConsts.SDBuzzerState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SDBuzzerState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SDBuzzerState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.SDBuzzerState.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SDBuzzerState" class**
Remark		**※ Reference (3.2.SDTagBuzzerState)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SD_GetLEDEnableState		
Declare		**public int SD_GetLEDEnableState()**
Description		Gets the LED enable state of the SLED
Parameter		void
Return	**Reader**	**Success :** Value of the buzzer state **(DISABLE(0), ENABLE(1))** **Serial Error** SDConsts.SDBuzzerState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.SDBuzzerState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.SDBuzzerState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.SDBuzzerState.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "SDBuzzerState" class.**
	BTReader	**Success :** Value of the buzzer state **(DISABLE (0), ENABLE (1))** **Enabled Error** : SDConsts.SDBuzzerState.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SDBuzzerState.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SDBuzzerState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SDBuzzerState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SDBuzzerState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.SDBuzzerState.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SDBuzzerState" class**
Remark		**※ Reference (3.2.SDLEDState)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SD_Wakeup		
Declare		**public int SD_Wakeup()**
Description		Wakes-up the SLED
Parameter		void
Reader Return		**Success :** Value of the SLED state **(SLEEP(0) , WAKEUP(1))** [Auto-update message from SLED] **SDConsts.SDCmdMsg** - SLED_WAKEUP = 47 **Condition Error** SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Other Error** SDConsts.SDResult.***OTHER_ERROR*** = -1 *** Can receive other error constant of "SDResult" class.**
Remark		**※ Reference (3.2.SDState)** **In case of this API, Run time during about 800 milliseconds is required. After the**

		operation completed, it sends related callback message. ***(SLED_WAKEUP(47))*** **※ This API is only for Serial interface(Reader)**

SD_GetChargeState		
Declare		**public int SD_GetChargeState()**
Description		Gets the charge state of the SLED
Parameter		void
Return	**Reader**	**Success :** Value of the charge state **(Off(0), On(1))** **Serial Error** SDConsts.SDChargeState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.SDChargeState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.SDChargeState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.SDChargeState.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "SDChargeState" class.**
	BTReader	**Success :** Value of the charge state **(Off(0), On(1))** **Enabled Error** : SDConsts.SDChargeState.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SDChargeState.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SDChargeState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SDChargeState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SDChargeState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.SDChargeState.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SDChargeState" class**
Remark		**※ Reference (3.2.SDChargeState)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SD_GetSerialNumber		
Declare		**public int SD_GetSerialNumber()**
Description		Gets the serial number of the SLED
Parameter		void
Return	**Reader**	**Success :** Value of the serial number **Serial Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** = "Error" *** Can receive other error constant of "SDResult" class.**

	BTReader	**Success :** Value of the serial number **Enabled Error** : SDConsts.SDResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : "Error" **Block State Error** : SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.SDResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SDResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SD_GetHostSerialNumber

Declare		**public String SD_GetHostSerialNumber()**
Description		Gets the serial number of Host Device.
Parameter		void
Return	**Reader**	**Success :** Value of the serial number **Fail : NULL**
	BTReader	
Remark		**※ Support only Serial**

SD_UpdateSLEDFirmware

Declare		**public int SD_UpdateSLEDFirmware(String filepath)**
Description		Updates the firmware of the SLED
Parameter		**filepath** - File path for SLED firmware update
Return	**Reader**	**Success** Constants.SDResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.SDCmdMsg** - UPDATE_SD_FW_START = 48 - UPDATE_SD_FW = 49 - UPDATE_SD_FW_END = 50 **File path Error** SDConsts.SDResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4

		Connected Error SDConsts.SDResult.***SD_NOT_CONNECTED*** = -5 **Charging state Error** SDConsts.SDResult.***CHARGING_STATE_ERROR*** = -14 *** Can receive other error constant of "SDResult" class.**
	BTReader	**Success** Constants.SDResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.SDCmdMsg** - UPDATE_SD_FW_START = 48 - UPDATE_SD_FW = 49 - UPDATE_SD_FW_END = 50 **Enabled Error** : SDConsts.SDResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SDResult.***SD_NOT_CONNECTED*** = -5 **File Path Error** : SDConsts.SDResult.***ARGUMENT_ERROR*** = -3 **Block State Error** : SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Charge Error** : SDConsts.SDResult.***CHARGING_STATE_ERROR*** = -14 **Hotswap Error** : SDConsts.SDResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SDResult" class.**
Remark		**Do update unconditionally** **In case of this API, Run time during about 90 seconds is required.** **It sends related callback message*(UPDATE_SD_FW_START(48) → UPDATE_SD_FW(49) → UPDATE_SD_FW_END(50))* at the beginning and the end.** **If it start FW's update, we recommend that do not be call any other cmd.** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH** - **android.Manifest.permission.WRITE_EXTERNAL_STORAGE** - **android.Manifest.permission.READ_EXTERNAL_STORAGE** **※ [(Reader) Requires permission]** - **android.Manifest.permission.WRITE_EXTERNAL_STORAGE** - **android.Manifest.permission.READ_EXTERNAL_STORAGE**

		SD_UpdateSLEDFirmware
Declare		**public int SD_UpdateSLEDFirmware(Uri uri)**
Description		Updates the firmware of the SLED
Parameter		**uri** - File path for SLED firmware update
Return	**Reader**	**Success** Constants.SDResult.***SUCCESS*** = 0

		[Auto-update message from SLED] **SDConsts.SDCmdMsg** - UPDATE_SD_FW_START = 48 - UPDATE_SD_FW = 49 - UPDATE_SD_FW_END = 50 **File path Error** SDConsts.SDResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.SDResult.***SD_NOT_CONNECTED*** = -5 **Charging state Error** SDConsts.SDResult.***CHARGING_STATE_ERROR*** = -14 *** Can receive other error constant of "SDResult" class.**
	BTReader	**Success** Constants.SDResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.SDCmdMsg** - UPDATE_SD_FW_START = 48 - UPDATE_SD_FW = 49 - UPDATE_SD_FW_END = 50 **Enabled Error** : SDConsts.SDResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SDResult.***SD_NOT_CONNECTED*** = -5 **File Path Error** : SDConsts.SDResult.***ARGUMENT_ERROR*** = -3 **Block State Error** : SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Charge Error** : SDConsts.SDResult.***CHARGING_STATE_ERROR*** = -14 **Hotswap Error** : SDConsts.SDResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SDResult" class.**
Remark		**Do update unconditionally** **In case of this API, Run time during about 90 seconds is required.** **It sends related callback message*(UPDATE_SD_FW_START(48) → UPDATE_SD_FW(49) → UPDATE_SD_FW_END(50))* at the beginning and the end.** **If it start FW's update, we recommend that do not be call any other cmd.** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH** - **android.Manifest.permission.WRITE_EXTERNAL_STORAGE** - **android.Manifest.permission.READ_EXTERNAL_STORAGE** **※ [(Reader) Requires permission]** - **android.Manifest.permission.WRITE_EXTERNAL_STORAGE**

	- **android.Manifest.permission.READ_EXTERNAL_STORAGE**

	~~SD_UpdateSLEDBootloader~~
~~Declare~~	**~~public int SD_UpdateSLEDBootloader(String filepath)~~**
~~Description~~	~~Updates the boot loader of the SLED~~ ~~Do update unconditionally~~
~~Parameter~~	**~~filepath~~**
~~Reader Return~~	**~~Success~~** ~~Constants.SDResult.***SUCCESS*** = 0~~ **~~File path Error~~** ~~SDConsts.SDResult.***ARGUMENT_ERROR*** = -3~~ **~~Serial Error~~** ~~SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4~~ **~~Condition Error~~** ~~SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7~~ **~~Command State Error~~** ~~SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4~~ **~~Connected Error~~** ~~SDConsts.SDResult.***SD_NOT_CONNECTED*** = -5~~ **~~Charging state Error~~** ~~SDConsts.SDResult.***CHARGING_STATE_ERROR*** = -14~~ **~~* Can receive other error constant of "SDResult" class.~~**
Remark	**~~※ [(BTReader) Requires permission]~~** ~~-~~ **~~android.Manifest.permission.BLUETOOTH~~** ~~-~~ **~~android.Manifest.permission.WRITE_EXTERNAL_STORAGE~~** - **~~android.Manifest.permission.READ_EXTERNAL_STORAGE~~** **~~※ [(Reader) Requires permission]~~** ~~-~~ **~~android.Manifest.permission.WRITE_EXTERNAL_STORAGE~~** - **~~android.Manifest.permission.READ_EXTERNAL_STORAGE~~**

	SD_SmartUpdateSLEDFirmware
Declare	**public int SD_SmartUpdateSLEDFirmware(String filepath)**
Description	Updates the firmware of the SLED smartly
Parameter	**filepath** - File path for SLED firmware update
Return **Reader**	**Success** Constants.SDResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.SDCmdMsg** - UPDATE_SD_FW_START = 48 - UPDATE_SD_FW = 49 - UPDATE_SD_FW_END = 50 **File path Error** SDConsts.SDResult.***ARGUMENT_ERROR*** = -3

		Serial Error SDConsts.SDResult.*OTHER_CMD_RUNNING_ERROR* = -4 **Condition Error** SDConsts.SDResult.*READER_OR_SERIAL_STATUS_ERROR* = -7 **Command State Error** SDConsts.SDResult.*OTHER_CMD_RUNNING_ERROR* = -4 **Connected Error** SDConsts.SDResult.*SD_NOT_CONNECTED* = -5 **Charging state Error** SDConsts.SDResult.*CHARGING_STATE_ERROR* = -14 **Not Supported Error** SDConsts.SDResult.*NOT_SUPPORTED_API* = -36 *** Can receive other error constant of "SDResult" class.**
	BTReader	**Success** Constants.SDResult.*SUCCESS* = 0 [Auto-update message from SLED] **SDConsts.SDCmdMsg** - UPDATE_SD_FW_START = 48 - UPDATE_SD_FW = 49 - UPDATE_SD_FW_END = 50 **Enabled Error** : SDConsts.SDResult.*BLUETOOTH_NOT_ENABLED* = -15 **Connected Error** : SDConsts.SDResult.*SD_NOT_CONNECTED* = -5 **File Path Error** : SDConsts.SDResult.*ARGUMENT_ERROR* = -3 **Block State Error** : SDConsts.SDResult.*OTHER_CMD_RUNNING_ERROR* = -4 **Condition Error** : SDConsts.SDResult.*READER_OR_SERIAL_STATUS_ERROR* = -7 **Command State Error** : SDConsts.SDResult.*OTHER_CMD_RUNNING_ERROR* = -4 **Charge Error** : SDConsts.SDResult.*CHARGING_STATE_ERROR* = -14 **Hotswap Error** : SDConsts.SDResult.*ERROR_HOTSWAP_STATE* = -37 *** Can receive other error constant of "SDResult" class.**
Remark		**Do update If new firmware binary is newest than current version** **In case of this API, Run time during about 90 seconds is required.** **It sends related callback message***(UPDATE_SD_FW_START(48) → UPDATE_SD_FW(49) → UPDATE_SD_FW_END(50))* **at the beginning and the end.** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH** - **android.Manifest.permission.WRITE_EXTERNAL_STORAGE** - **android.Manifest.permission.READ_EXTERNAL_STORAGE** **※ [(Reader) Requires permission]** - **android.Manifest.permission.WRITE_EXTERNAL_STORAGE** - **android.Manifest.permission.READ_EXTERNAL_STORAGE**

SD_SetModeKeyEnable

Declare	**public int SD_SetModeKeyEnable(int SDModeKeyState)**
Description	Sets the mode key enable state of the SLED
Parameter	SDModeKeyState - Enable : 1

		- Disable : 0
Return	**Reader**	**Success** Constants.SDResult.***SUCCESS*** = 0 **Mode Key Range Error** SDConsts.SDResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.SDResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "SDResult" class.**
	BTReader	**Success** Constants.SDResult.***SUCCESS*** = 0 **Range Error** : SDConsts.SDResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** : SDConsts.SDResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SDResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.SDResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SDResult" class.**
Remark		**In case of "Disable" state in this API, user can control barcode beam through BC_SetTriggerState API.** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		SD_GetModeKeyEnableState
Declare		**public int SD_GetModeKeyEnableState()**
Description		Gets the mode key enable state of the SLED
Parameter		void
Return	**Reader**	**Success :** Value of the key enable state **(Disable(0), Enable(1))** **Serial Error** SDConsts.SDModeKeyState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.SDModeKeyState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.SDModeKeyState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.SDModeKeyState.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "SDModeKeyState" class.**
	BTReader	**Success :** Value of the key enable state **(Disable(0), Enable(1))** **Enabled Error** : SDConsts.SDModeKeyState.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SDModeKeyState.***SD_NOT_CONNECTED*** = -5

		Block State Error : SDConsts.SDModeKeyState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SDModeKeyState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SDModeKeyState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.SDModeKeyState.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SDModeKeyState" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SD_SetTriggerKeyEnable

Declare		**public int SD_SetTriggerKeyEnable(int SDTriggerKeyState)**
Description		Sets the trigger key event enable state of the SLED
Parameter		SDTriggerKeyState - Enable : 1 - Disable : 0
Return	**Reader**	**Success** Constants.SDResult.***SUCCESS*** = 0 **Trigger Key Range Error** SDConsts.SDResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.SDResult.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "SDResult" class.**
	BTReader	**Success** Constants.SDResult.***SUCCESS*** = 0 **Range Error** : SDConsts.SDResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** : SDConsts.SDResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SDResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.SDResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SDResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SD_GetTriggerKeyEnableState

Declare		**public int SD_GetTriggerKeyEnableState()**

Description		Gets the trigger key event enable state of the SLED
Parameter		void
Return	**Reader**	**Success :** Value of the key enable state **(Disable(0), Enable(1))** **Serial Error** SDConsts.SDTriggerKeyState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.SDTriggerKeyState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.SDTriggerKeyState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.SDTriggerKeyState.***SD_NOT_CONNECTED*** = -5 *** Can receive other error constant of "SDTriggerKeyState" class.**
	BTReader	**Success :** Value of the key enable state **(Disable(0), Enable(1))** **Enabled Error** : SDConsts.SDTriggerKeyState.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SDTriggerKeyState.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SDTriggerKeyState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SDTriggerKeyState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SDTriggerKeyState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.SDTriggerKeyState.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SDTriggerKeyState" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SD_SetBTName

Declare		**public String SD_SetBTName(String SledBluetoothDeviceName)**
Description		Set Bluetooth name of SLED
Parameter		SledBluetoothDeviceName - Bluetooth name of SLED
Return	**Reader**	**Success :** Constants.SDResult.***SUCCESS*** = 0 **Enable Error** SDConsts.SDResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** SDConsts.SDResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.SDResult.***READER_OR_COM_INTERFACE_STATUS_ERROR*** = -7 **Command State Error** SDConsts. SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 *** Can receive other error constant of "SDResult" class.**
	BTReader	**Success :** Constants.SDResult.***SUCCESS*** = 0 **Enabled Error** : SDConsts.SDResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SDResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4

		Condition Error : SDConsts.SDResult.*READER_OR_SERIAL_STATUS_ERROR* = -7 **Command State Error** : SDConsts.SDResult.*OTHER_CMD_RUNNING_ERROR* = -4 **Hotswap Error** : SDConsts.SDResult.*ERROR_HOTSWAP_STATE* = -37 *** Can receive other error constant of “SDResult” class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SD_GetBTName		
Declare		**public String SD_GetBTName()**
Description		Get Bluetooth name of SLED
Parameter		void
Return	**Reader**	**Success :** Bluetooth name of SLED **Enable Error** SDConsts.SDResult.*BLUETOOTH_NOT_ENABLED* = -15 **Connected Error** = “Error” **Block State Error** SDConsts.SDResult.*OTHER_CMD_RUNNING_ERROR* = -4 **Condition Error** SDConsts.SDResult.*READER_OR_COM_INTERFACE_STATUS_ERROR* = -7 **Command State Error** SDConsts. SDResult.*OTHER_CMD_RUNNING_ERROR* = -4 *** Can receive other error constant of “SDResult” class.**
	BTReader	**Success :** Bluetooth name of SLED **Enabled Error** : SDConsts.SDResult.*BLUETOOTH_NOT_ENABLED* = -15 **Connected Error** : “Error” **Block State Error** : SDConsts.SDResult.*OTHER_CMD_RUNNING_ERROR* = -4 **Condition Error** : SDConsts.SDResult.*READER_OR_SERIAL_STATUS_ERROR* = -7 **Command State Error** : SDConsts.SDResult.*OTHER_CMD_RUNNING_ERROR* = -4 **Hotswap Error** : SDConsts.SDResult.*ERROR_HOTSWAP_STATE* = -37 *** Can receive other error constant of “SDResult” class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SD_GetBTVersion		
Declare		**public String SD_GetBTVersion()**
Description		Gets the bluetooth's firmware version of SLED
Parameter		void
Return	**Reader**	**Success :** Version of the Bluetooth firmware

		Enable Error SDConsts.SDResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** = “Error” **Block State Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.SDResult.***READER_OR_COM_INTERFACE_STATUS_ERROR*** = -7 **Command State Error** SDConsts. SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 *** Can receive other error constant of “SDResult” class.**
	BTReader	**Success :** Version of the Bluetooth firmware **Enabled Error** : SDConsts.SDResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : “Error” **Block State Error** : SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.SDResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of “SDResult” class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		SD_ResetConfiguration
Declare		**public int SD_ResetConfiguration()**
Description		Resets the setting values of the SLED
Parameter		void
Return	**Reader**	**Success :** Reset SLED Default - Buzzer volume = 1(Mid) - Buzzer Enable : True - Sleep Timeout : 30 seconds - BT NAME : RFR-XXXXX (Works only with device embedded bluetooth) - Batch Data : All Clean - Trigger Mode : RFID - Mode Key/Trigger Key Enable : True **Serial Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.SDResult.***SD_NOT_CONNECTED*** = -5 **Not Supported Error** SDConsts.SDResult.***NOT_SUPPORTED_API*** = -36 *** Can receive other error constant of “SDResult” class.**

	BTReader	**Success :** Reset SLED Default - Buzzer volume = 1(Mid) - Buzzer Enable : True - Sleep Timeout : 30 seconds - BT NAME : RFR-XXXXX (Works only with device embedded Bluetooth) - Batch Data : All Clean - Trigger Mode : RFID - Mode Key/Trigger Key Enable : True **Enabled Error** : SDConsts.SDResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SDResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts.SDResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SDResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		SD_UpdateSLEDFirmwareAndDYN
Declare		**public int SD_UpdateSLEDFirmwareAndDYN(String filepath)**
Description		Updates the firmware of the SLED/Dynamic Profile
Parameter		**filepath** - File path for SLED firmware update
Return	**Reader**	**Success** Constants.SDResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.SDCmdMsg** - UPDATE_SD_FW_START = 48 - UPDATE_SD_FW = 49 - UPDATE_SD_FW_END = 50 - **SDConsts.RFCmdMsg** - UPDATE_RF_DYN_MAC_START = 53 - UPDATE_RF_DYN_MAC_FW = 54 - UPDATE_RF_DYN_MAC_END = 55 **File path Error** SDConsts.SDResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4

		Condition Error SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.SDResult.***SD_NOT_CONNECTED*** = -5 **Charging state Error** SDConsts.SDResult.***CHARGING_STATE_ERROR*** = -14 *** Can receive other error constant of "SDResult" class.**
	BTReader	**Success** Constants.SDResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.SDCmdMsg** - UPDATE_SD_FW_START = 48 - UPDATE_SD_FW = 49 - UPDATE_SD_FW_END = 50 **Enabled Error** : SDConsts.SDResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SDResult.***SD_NOT_CONNECTED*** = -5 **File Path Error** : SDConsts.SDResult.***ARGUMENT_ERROR*** = -3 **Block State Error** : SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Charge Error** : SDConsts.SDResult.***CHARGING_STATE_ERROR*** = -14 **Hotswap Error** : SDConsts.SDResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SDResult" class.**
Remark		**Do update unconditionally** **In case of this API, Run time during about 180 seconds is required.** **First for Dynamic Profile, It sends related callback message*(UPDATE_RF_DYN_MAC_START (53) → UPDATE_RF_DYN_MAC_FW (54) → UPDATE_RF_DYN_MAC_END (55))* at the beginning and the end.** **Second for SLED Firmware, It sends related callback message*(UPDATE_SD_FW_START(48) → UPDATE_SD_FW(49) → UPDATE_SD_FW_END(50))* at the beginning and the end.** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH** - **android.Manifest.permission.WRITE_EXTERNAL_STORAGE** - **android.Manifest.permission.READ_EXTERNAL_STORAGE** **※ [(Reader) Requires permission]** - **android.Manifest.permission.WRITE_EXTERNAL_STORAGE** - **android.Manifest.permission.READ_EXTERNAL_STORAGE**

SD_GetSmartBatterySerial		
Declare		**public String SD_GetSmartBatterySerial()**
Description		Get the smart battery serial
Parameter		void
Return	**Reader**	**Success :** Value of smart battery serial **Serial Error** SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts. SDBatteryState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** = "Error" **Charging state Error** SDConsts. SDBatteryState.***CHARGING_STATE_ERROR*** = -14 *** Can receive other error constant of "SDBatteryState" class.**
	BTReader	**Success :** Value of smart battery serial **Enabled Error** : SDConsts. SDBatteryState.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : "Error" **Block State Error** : SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts. SDBatteryState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts. SDBatteryState.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SDBatteryState" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SD_GetSmartBatteryStatus		
Declare		**public String SD_GetSmartBatteryStatus()**
Description		Get the smart battery status
Parameter		void
Return	**Reader**	**Success :** Value of smart battery status **Serial Error** SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts. SDBatteryState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** = "Error" **Charging state Error** SDConsts. SDBatteryState.***CHARGING_STATE_ERROR*** = -14 *** Can receive other error constant of "SDBatteryState" class.**
	BTReader	**Success :** Value of smart battery status

		Enabled Error : SDConsts. SDBatteryState.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : "Error" **Block State Error** : SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts. SDBatteryState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts. SDBatteryState.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SDBatteryState" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SD_GetSmartBatteryVoltage		
Declare		**public String SD_GetSmartBatteryVoltage()**
Description		Get the smart battery voltage
Parameter		void
Return	**Reader**	**Success :** Value of smart battery voltage **Serial Error** SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts. SDBatteryState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** = "Error" **Charging state Error** SDConsts. SDBatteryState.***CHARGING_STATE_ERROR*** = -14 *** Can receive other error constant of "SDBatteryState" class.**
	BTReader	**Success :** Value of smart battery voltage **Enabled Error** : SDConsts. SDBatteryState.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : "Error" **Block State Error** : SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts. SDBatteryState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts. SDBatteryState.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SDBatteryState" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SD_GetSmartBatteryPresentStatus		
Declare		**public String SD_GetSmartBatteryPresentStatus()**
Description		Get the smart battery present status

Parameter		void
Return	**Reader**	**Success :** Value of smart battery present status **Serial Error** SDConsts. SDBatteryState.*OTHER_CMD_RUNNING_ERROR* = -4 **Condition Error** SDConsts. SDBatteryState.*READER_OR_SERIAL_STATUS_ERROR* = -7 **Command State Error** SDConsts. SDBatteryState.*OTHER_CMD_RUNNING_ERROR* = -4 **Connected Error** = “Error” **Charging state Error** SDConsts. SDBatteryState.*CHARGING_STATE_ERROR* = -14 *** Can receive other error constant of “SDBatteryState” class.**
	BTReader	**Success :** Value of smart battery present status **Enabled Error** : SDConsts. SDBatteryState.*BLUETOOTH_NOT_ENABLED* = -15 **Connected Error** : “Error” **Block State Error** : SDConsts. SDBatteryState.*OTHER_CMD_RUNNING_ERROR* = -4 **Condition Error** : SDConsts. SDBatteryState.*READER_OR_SERIAL_STATUS_ERROR* = -7 **Command State Error** : SDConsts. SDBatteryState.*OTHER_CMD_RUNNING_ERROR* = -4 **Hotswap Error** : SDConsts. SDBatteryState.*ERROR_HOTSWAP_STATE* = -37 *** Can receive other error constant of “SDBatteryState” class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SD_GetSmartBatteryLevel

Declare		**public String SD_GetSmartBatteryLevel()**
Description		Get the smart battery level
Parameter		void
Return	**Reader**	**Success :** Value of smart battery level **Serial Error** SDConsts. SDBatteryState.*OTHER_CMD_RUNNING_ERROR* = -4 **Condition Error** SDConsts. SDBatteryState.*READER_OR_SERIAL_STATUS_ERROR* = -7 **Command State Error** SDConsts. SDBatteryState.*OTHER_CMD_RUNNING_ERROR* = -4 **Connected Error** = “Error” **Charging state Error** SDConsts. SDBatteryState.*CHARGING_STATE_ERROR* = -14 *** Can receive other error constant of “SDBatteryState” class.**
	BTReader	**Success :** Value of smart battery level **Enabled Error** : SDConsts. SDBatteryState.*BLUETOOTH_NOT_ENABLED* = -15 **Connected Error** : “Error” **Block State Error** : SDConsts. SDBatteryState.*OTHER_CMD_RUNNING_ERROR* = -4

		Condition Error : SDConsts. SDBatteryState.*READER_OR_SERIAL_STATUS_ERROR* = -7 **Command State Error** : SDConsts. SDBatteryState.*OTHER_CMD_RUNNING_ERROR* = -4 **Hotswap Error** : SDConsts. SDBatteryState.*ERROR_HOTSWAP_STATE* = -37 *** Can receive other error constant of "SDBatteryState" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SD_GetSmartBatteryLifeTime		
Declare		**public String SD_GetSmartBatteryLifeTime()**
Description		Get the smart battery life time
Parameter		void
Return	**Reader**	**Success :** Value of smart battery life time **Serial Error** SDConsts. SDBatteryState.*OTHER_CMD_RUNNING_ERROR* = -4 **Condition Error** SDConsts. SDBatteryState.*READER_OR_SERIAL_STATUS_ERROR* = -7 **Command State Error** SDConsts. SDBatteryState.*OTHER_CMD_RUNNING_ERROR* = -4 **Connected Error** = "Error" **Charging state Error** SDConsts. SDBatteryState.*CHARGING_STATE_ERROR* = -14 *** Can receive other error constant of "SDBatteryState" class.**
	BTReader	**Success :** Value of smart battery life time **Enabled Error** : SDConsts. SDBatteryState.*BLUETOOTH_NOT_ENABLED* = -15 **Connected Error** : "Error" **Block State Error** : SDConsts. SDBatteryState.*OTHER_CMD_RUNNING_ERROR* = -4 **Condition Error** : SDConsts. SDBatteryState.*READER_OR_SERIAL_STATUS_ERROR* = -7 **Command State Error** : SDConsts. SDBatteryState.*OTHER_CMD_RUNNING_ERROR* = -4 **Hotswap Error** : SDConsts. SDBatteryState.*ERROR_HOTSWAP_STATE* = -37 *** Can receive other error constant of "SDBatteryState" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SD_GetSmartBatteryHealth		
Declare		**public String SD_GetSmartBatteryHealth()**
Description		Get the smart battery health
Parameter		void
Return	**Reader**	**Success :** Value of smart battery health

		Serial Error SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts. SDBatteryState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** = "Error" **Charging state Error** SDConsts. SDBatteryState.***CHARGING_STATE_ERROR*** = -14 *** Can receive other error constant of "SDBatteryState" class.**
	BTReader	**Success :** Value of smart battery health **Enabled Error** : SDConsts. SDBatteryState.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : "Error" **Block State Error** : SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts. SDBatteryState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts. SDBatteryState.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SDBatteryState" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SD_GetSmartBatteryTemperature

Declare		**public String SD_GetSmartBatteryTemperature()**
Description		Get the smart battery temperature
Parameter		void
Return	**Reader**	**Success :** Value of smart battery temperature **Serial Error** SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts. SDBatteryState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** = "Error" **Charging state Error** SDConsts. SDBatteryState.***CHARGING_STATE_ERROR*** = -14 *** Can receive other error constant of "SDBatteryState" class.**
	BTReader	**Success :** Value of smart battery temperature **Enabled Error** : SDConsts. SDBatteryState.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : "Error" **Block State Error** : SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts. SDBatteryState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts. SDBatteryState.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SDBatteryState" class.**

Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		SD_GetSmartBatteryCycleCnt
Declare		**public String SD_GetSmartBatteryCycleCnt()**
Description		Get the smart battery cycle count
Parameter		void
Return	**Reader**	**Success :** Value of smart battery cycle count **Serial Error** SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts. SDBatteryState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** = “Error” **Charging state Error** SDConsts. SDBatteryState.***CHARGING_STATE_ERROR*** = -14 *** Can receive other error constant of “SDBatteryState” class.**
	BTReader	**Success :** Value of smart battery cycle count **Enabled Error** : SDConsts. SDBatteryState.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : “Error” **Block State Error** : SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts. SDBatteryState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts. SDBatteryState.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of “SDBatteryState” class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		SD_GetSmartBatteryCapacity
Declare		**public String SD_GetSmartBatteryCapacity()**
Description		Get the smart battery capacity
Parameter		void
Return	**Reader**	**Success :** Value of smart battery capacity **Serial Error** SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts. SDBatteryState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** = “Error”

		Charging state Error SDConsts. SDBatteryState.***CHARGING_STATE_ERROR*** = -14 *** Can receive other error constant of "SDBatteryState" class.**
	BTReader	**Success :** Value of smart battery capacity **Enabled Error** : SDConsts. SDBatteryState.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : "Error" **Block State Error** : SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts. SDBatteryState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts. SDBatteryState.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SDBatteryState" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SD_GetSmartBatteryCycleCnt		
Declare		**public String SD_GetSmartBatteryCycleCnt()**
Description		Get the smart battery cycle count
Parameter		void
Return	**Reader**	**Success :** Value of smart battery cycle count **Serial Error** SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts. SDBatteryState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** = "Error" **Charging state Error** SDConsts. SDBatteryState.***CHARGING_STATE_ERROR*** = -14 *** Can receive other error constant of "SDBatteryState" class.**
	BTReader	**Success :** Value of smart battery cycle count **Enabled Error** : SDConsts. SDBatteryState.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : "Error" **Block State Error** : SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts. SDBatteryState.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts. SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Error** : SDConsts. SDBatteryState.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SDBatteryState" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		SD_GetType
Declare		**public int SD_GetType()**
Description		Get the SLED Type
Parameter		void
Return	**Reader**	**Success :** Value of the Slde type **SDConsts.SLED_UNKOWN = 0** **SDConsts.SLED_INTERNAL_SLED = 1** **SDConsts.RFR900_EXTERNAL_SLED = 2** **SDConsts.RFR901_EXTERNAL_SLED = 3** **Serial Error** SDConsts.SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.SDBatteryState.***READER_OR_SERIAL_STATUS_ERROR***= -7 **Command State Error** SDConsts.SDBatteryState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** SDConsts.SDBatteryState.***SD_NOT_CONNECTED***= -5
	BTReader	**.**
Remark		**※ Support only Serial.**

■ SB APIs

SB_ResetBarcodeConfiguration		
Declare		**public int SB_ResetBarcodeConfiguration()**
Description		Resets bar code configuration
Parameter		void
Return	**Reader**	**Success :** Reset SLED Default **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Board Type Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 *** Can receive other error constant of "SBResult" class.**
	BTReader	**Success :** Reset SLED Default **Enabled Error** : SDConsts.SBResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Board Type Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 **Hotswap Error** : SDConsts.SBResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SBResult" class.**
Remark		**※ Reference barcode default configuration(3.6 Barcode parameters)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SB_EnableBarcodeSound		
Declare		**public int SB_EnableBarcodeSound(boolean enable)**
Description		Enables/Disables barcode read sound
Parameter		enable - True : Enable sound - False : Disable sound
Return	**Reader**	**Success :** SDConsts.SBResult.***SUCCESS*** = 0 **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5

		Block State Error : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 *** Can receive other error constant of "SBResult" class.**
	BTReader	**Success :** SDConsts.SBResult.***SUCCESS*** = 0 **Enabled Error** : SDConsts.SBResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 **Hotswap Error** : SDConsts.SBResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SBResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SB_GetBarcodeSoundState		
Declare		**public int SB_GetBarcodeSoundState()**
Description		Gets barcode read sound enable state
Parameter		void
Return	**Reader**	**Success :** Enable state of barcode read sound **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 *** Can receive other error constant of "SBResult" class.**
	BTReader	**Success :** Enable state of barcode read sound **Enabled Error** : SDConsts.SBResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 **Hotswap Error** : SDConsts.SBResult.***ERROR_HOTSWAP_STATE*** = -37

		* Can receive other error constant of “SBResult” class.
Remark		※ [(BTReader) Requires permission] - android.Manifest.permission.BLUETOOTH

		SB_SetBarcodeTriggerMode
Declare		**public int SB_SetBarcodeTriggerMode(int SBBarcodeTriggerMode)**
Description		Sets barcode scan mode
Parameter		SBBarcodeTriggerMode **(0 ~ 3)** - SDConstsBT.SBBarcodeTriggerMode.LEVEL = 0; - SDConstsBT.SBBarcodeTriggerMode.PULSE = 1; - SDConstsBT.SBBarcodeTriggerMode.EDGE = 2; - SDConstsBT.SBBarcodeTriggerMode.AUTOSTAND = 3;
Return	**Reader**	**Success :** SDConsts.SBResult.***SUCCESS*** = 0 **Argument Error** : SDConsts.SBResult.***ARGUMENT_ERROR*** = -3 **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 *** Can receive other error constant of “SBResult” class.**
	BTReader	**Success :** SDConsts.SBResult.***SUCCESS*** = 0 **Argument Error** : SDConsts.SBResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** : SDConsts.SBResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 **Hotswap Error** : SDConsts.SBResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of “SBResult” class.**
Remark		※ [(BTReader) Requires permission] - android.Manifest.permission.BLUETOOTH

		SB_GetBarcodeTriggerMode
Declare		**public int SB_GetBarcodeTriggerMode()**

Description		Gets barcode scan mode
Parameter		void
Return	**Reader**	**Success :** Value of trigger mode(LEVEL(0)~AUTOSTAND(3)) **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 *** Can receive other error constant of "SBResult" class.**
	BTReader	**Success :** Value of trigger mode(LEVEL(0)~AUTOSTAND(3)) **Enabled Error** : SDConsts.SBResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 **Hotswap Error** : SDConsts.SBResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SBResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SB_EnableBarcodeScanner

Declare		**public int SB_EnableBarcodeScanner(boolean enable)**
Description		Permits bar code scanning or Prevents the operator from scanning bar codes.
Parameter		enable - True : Enable - False : Disable
Return	**Reader**	**Success :** SDConsts.SBResult.***SUCCESS*** = 0 **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 *** Can receive other error constant of "SBResult" class.**
	BTReader	**Success :** SDConsts.SBResult.***SUCCESS*** = 0

		Enabled Error : SDConsts.SBResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 **Hotswap Error** : SDConsts.SBResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SBResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SB_EnableAim

Declare		**public int SB_EnableAim(boolean enable)**
Description		Activates/ Deactivates aim pattern.
Parameter		enable - True : Activate - False : Deactivate
Return	**Reader**	**Success :** SDConsts.SBResult.***SUCCESS*** = 0 **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 *** Can receive other error constant of "SBResult" class.**
	BTReader	**Success :** SDConsts.SBResult.***SUCCESS*** = 0 **Enabled Error** : SDConsts.SBResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 **Hotswap Error** : SDConsts.SBResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SBResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		SB_EnableIllumination
Declare		**public int SB_EnableIllumination(boolean enable)**
Description		Activates/ Deactivates Illumination
Parameter		enable - True : Activate - False : Deactivate
Return	**Reader**	**Success :** SDConsts.SBResult.***SUCCESS*** = 0 **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 *** Can receive other error constant of "SBResult" class.**
	BTReader	**Success :** SDConsts.SBResult.***SUCCESS*** = 0 **Enabled Error** : SDConsts.SBResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 **Hotswap Error** : SDConsts.SBResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SBResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		SB_EnableIllumination
Declare		**public int SB_EnableIllumination(boolean enable,byte[] imageData)**
Description		Activates/ Deactivates Illumination
Parameter		enable - True : Activate - False : Deactivate imageData - Null ~ 251 bytes, SB_ILLUMINATION_DATA_MAX_SIZE)
Return	**Reader**	**Success :** SDConsts.SBResult.***SUCCESS*** = 0 **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4

		Condition Error : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 *** Can receive other error constant of "SBResult" class.**
	BTReader	**Success :** SDConsts.SBResult.***SUCCESS*** = 0 **Enabled Error** : SDConsts.SBResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 **Hotswap Error** : SDConsts.SBResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SBResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SB_GetRevision		
Declare		**public int SB_GetRevision()**
Description		Gets the decoder's Revision value
Parameter		void
Return	**Reader**	**Success :** Value of decoder's revision **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 *** Can receive other error constant of "SBResult" class.**
	BTReader	**Success :** Value of decoder's revision **Enabled Error** : SDConsts.SBResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 **Hotswap Error** : SDConsts.SBResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SBResult" class.**

Remark	**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

~~**SD_StartScanSLEDBarcode**~~	
~~**Declare**~~	~~**public int SD_StartScanSLEDBarcode(boolean start)**~~
~~**Description**~~	~~Starts/Stops barcode on SLED~~
~~**Parameter**~~	~~start~~ - ~~True : Start~~ - ~~False : Stop~~
~~**Reader Return**~~	~~**Success :** SDConsts.SDResult.***SUCCESS*** = 0~~ ~~**Serial Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4~~ ~~**Condition Error** SDConsts.SDResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7~~ ~~**Command State Error** SDConsts.SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4~~ ~~**Mode Error** SDConsts.SDResult.***MODE_ERROR*** = -6~~ ~~**Not Supported Error** SDConsts.SDResult.***NOT_SUPPORTED_API*** = -36~~ ~~**Connected Error** SDConsts.SDResult.***SD_NOT_CONNECTED*** = -5~~ ~~*** Can receive other error constant of "SDResult" class.**~~
~~**Remark**~~	~~**Not supported without barcode on SLED**~~ **※ This API is deprecated, use SB_StartScan**

SB_StartScan		
Declare		**public int SB_StartScan(boolean start)**
Description		Tells decoder to attempt to decode a bar code or Tells decoder to abort a decode attempt
Parameter		start - True : Start - False : Stop
Return	**Reader**	**Success :** SDConsts.SBResult.***SUCCESS*** = 0 **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 *** Can receive other error constant of "SBResult" class.**
	BTReader	**Success :** SDConsts.SBResult.***SUCCESS*** = 0

		Enabled Error : SDConsts.SBResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 **Hotswap Error** : SDConsts.SBResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SBResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

		SB_GetParamValue
Declare		**public int SB_GetParamValue(int SBParam)**
Description		Requests values of certain parameters.
Parameter		SBParam - Barcode parameters
Return	**Reader**	**Success :** Value of barcode parameters **Argument Error** : SDConsts.SBResult.***ARGUMENT_ERROR*** = -3 **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 *** Can receive other error constant of "SBResult" class.**
	BTReader	**Success :** Value of barcode parameters **Argument Error** : SDConsts.SBResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** : SDConsts.SBResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 **Hotswap Error** : SDConsts.SBResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SBResult" class.**
Remark		**※Reference (3.6 Barcode parameters)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SB_SetParamValue		
Declare		**public int SB_GetParamValue(int SBParam, int paramData)**
Description		Set values of certain parameters.
Parameter		SBParam - Barcode parameters paramData - Barcode parameters value
Return	**Reader**	**Success :** SDConsts.SBResult.***SUCCESS*** = 0 **Argument Error** : SDConsts.SBResult.***ARGUMENT_ERROR*** = -3 **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 *** Can receive other error constant of "SBResult" class.**
	BTReader	**Success :** SDConsts.SBResult.***SUCCESS*** = 0 **Argument Error** : SDConsts.SBResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** : SDConsts.SBResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 **Hotswap Error** : SDConsts.SBResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SBResult" class.**
Remark		**※Reference (3.6 Barcode parameters)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SB_SetBarcodePresetValue		
Declare		**public int SB_SetBarcodePresetValue(int SBPresetType, int presetData)**
Description		Set values of certain parameters.
Parameter		SBPresetType **(0 ~ 3)** - SDConsts.SBPresetType.PREFIX = 0

		- SDConsts.SBPresetType.SUFFIX = 1 - SDConsts.SBPresetType.PREAMBLE = 2 - SDConsts.SBPresetType.POSTAMBLE = 3 presetData - Preset data (Max length : SDConstsBT.SB_PRESET_VALUE_MAX_LENGTH)
Return	**Reader**	**Success :** SDConsts.SBResult.***SUCCESS*** = 0 **Argument Error** : SDConsts.SBResult.***ARGUMENT_ERROR*** = -3 **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 *** Can receive other error constant of "SBResult" class.**
	BTReader	**Success :** SDConsts.SBResult.***SUCCESS*** = 0 **Argument Error** : SDConsts.SBResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** : SDConsts.SBResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 **Hotswap Error** : SDConsts.SBResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SBResult" class.**
Remark		**※ [(BTReader) Requires permission]** **- android.Manifest.permission.BLUETOOTH**

		SB_GetBarcodePresetValue
Declare		**public int SB_GetBarcodePresetValue(int SBPresetType)**
Description		Gets the (prefix, suffix, preamble, postamble) data
Parameter		SBPresetType **(0 ~ 3)** - SDConsts.SBPresetType.PREFIX = 0 - SDConsts.SBPresetType.SUFFIX = 1 - SDConsts.SBPresetType.PREAMBLE = 2 - SDConsts.SBPresetType.POSTAMBLE = 3
Return	**Reader**	**Success :** Value of barcode preset **Argument Error** : SDConsts.SBResult.***ARGUMENT_ERROR*** = -3

		Connected Error : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 *** Can receive other error constant of "SBResult" class.**
	BTReader	**Success :** Value of barcode preset **Argument Error** : SDConsts.SBResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** : SDConsts.SBResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 **Hotswap Error** : SDConsts.SBResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "SBResult" class.**
Remark		**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

SB_GetSupportedDevicesInfo		
Declare		**public int SB_GetSupportedDevicesInfo ()**
Description		Check whether the device supports barcode
Parameter		- void
Return	**Reader**	**Success :** Supported barcode = 0 **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** : SDConsts.SDResult.***SD_NOT_CONNECTED*** = -5 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 *** Can receive other error constant of "SBResult" class.**
	BTReader	**Success :** Supported barcode = 0 **Enabled Error** : SDConsts.SBResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Connected Error** : SDConsts.SBResult.***SD_NOT_CONNECTED*** = -5 **Block State Error** : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.SBResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7

	Command State Error : SDConsts.SBResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SBResult.***NOT_SUPPORTED_API*** = -36 **Other Error :** SDConsts.SBResult.***OTHER_ERROR*** = -1 *** Can receive other error constant of "SBResult" class.**
Remark	**※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

■ BC APIs

		BC_SetTriggerState
Declare		**public int BC_SetTriggerState(Boolean isPress)**
Description		Sets the barcode trigger state
Parameter		**isPress** - True : Pressed - False : Non-pressed
Return	**Reader**	**Success** SDConsts.BCResult.***SUCCESS*** = 0 [Auto-update message from SLED] **SDConsts.SDCmdMsg** - TRIGGER_PRESSED = 41 - TRIGGER_RELEASED = 42 **Serial Error** SDConsts.BCResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.BCResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.BCResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Mode Error** SDConsts.BCResult.***MODE_ERROR*** = -6 **Low Battery Error** SDConsts.BCResult.***LOW_BATTERY*** = -12 *** Can receive other error constant of “BCResult” class.**
	BTReader	**Success** SDConsts.BCResult.***SUCCESS*** = 0 SDConsts.BCResult.***BARCODE_NOT_ACTIVE*** = -35 [Auto-update message from SLED] **SDConsts.SDCmdMsg** - TRIGGER_PRESSED = 41 - TRIGGER_RELEASED = 42 **Battery Error** : SDConsts.BCResult.***LOW_BATTERY*** = -12 **Block State Error** : SDConsts.BCResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.BCResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.BCResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Connected Error** : SDConsts.BCResult.***MODE_ERROR*** = -6 **Hotswap Error** : SDConsts.BCResult.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of “BCResult” class.**
Remark		**Starts/Stops the barcode scan on Bluebird Android Device with barcode** **(Not Supported on other devices)** **※ [(BTReader) Requires permission]** - **android.Manifest.permission.BLUETOOTH**

BC_PauseBarcode

Declare		**public int BC_PauseBarcode()**
Description		Sets the barcode state to pause state
Parameter		void
Return	**Reader**	**Success** SDConsts.BCResult.***SUCCESS*** = 0 SDConsts.BCResult.***ALREADY_PAUSE***= -34 **Serial Error** SDConsts.BCResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.BCResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.BCResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Low Battery Error** SDConsts.BCResult.***LOW_BATTERY*** = -12 *** Can receive other error constant of "BCResult" class.**
	BTReader	**Success** SDConsts.BCResult.***SUCCESS*** = 0 SDConsts.BCResult.***ALREADY_PAUSE***= -34 **Battery Error** : SDConsts.BCResult.***LOW_BATTERY*** = -12 **Block State Error** : SDConsts.BCResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.BCResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.BCResult.***OTHER_CMD_RUNNING_ERROR*** = -4 *** Can receive other error constant of "BCResult" class.**
Remark		**※ Reference (3.10 Barcode Lifecycle)** **Support only Bluebird Android Device with barcode** **(Not Supported on other devices)**

BC_ResumeBarcode

Declare		**public int BC_ResumeBarcode()**
Description		In the case of pause state on barcode, changes state to resume state
Parameter		void
Return	**Reader**	**Success** SDConsts.BCResult.***SUCCESS*** = 0 SDConsts.BCResult.***ALREADY_RESUME***= -33 **Serial Error** SDConsts.BCResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.BCResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.BCResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Low Battery Error** SDConsts.BCResult.***LOW_BATTERY*** = -12 *** Can receive other error constant of "BCResult" class.**
	BTReader	**Success** SDConsts.BCResult.***SUCCESS*** = 0

		SDConsts.BCResult.***ALREADY_RESUME***= -33 **Battery Error** : SDConsts.BCResult.***LOW_BATTERY*** = -12 **Block State Error** : SDConsts.BCResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** : SDConsts.BCResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** : SDConsts.BCResult.***OTHER_CMD_RUNNING_ERROR*** = -4 *** Can receive other error constant of "BCResult" class.**
Remark		**This API waits for a certain time(500ms, fixed time) to resume barcode** **※ Reference (3.10 Barcode Lifecycle)** **Support only Bluebird Android Device with barcode** **(Not Supported on other devices)**

BC_ResumeBarcode

Declare		**public void BC_ResumeBarcode(BCResumeListener resumeListener)**
Description		In the case of pause state on barcode, changes state to resume state
Parameter		resumeListener - The listener of result (0 : Success / -36 : Not Support)
Return	**Reader**	void
	BTReader	void
Remark		**This Apis notify the result of resuming via listener at once** **※ Reference (3.10 Barcode Lifecycle)** **Support only Bluebird Android Device with barcode** **(Not Supported on other devices)**

BC_GetBarcodeState

Declare		**public int BC_GetBarcodeState()**
Description		Gets the state of barcode(Active, Pause, Not active)
Parameter		void
Return	**Reader**	**Success** SDConsts.BCState.***ACTIVE*** = 0 SDConsts.BCState.***PAUSED*** = 1 SDConsts.BCState. ***NOT_ACTIVE*** = 2 **Low Battery Error** SDConsts.BCResult.***LOW_BATTERY*** = -12
	BTReader	
Remark		**Support only Bluebird Android Device with barcode** **(Not Supported on other devices)**

		BC_SetBarcodeKeyFormat
Declare		**public int BC_SetBarcodeKeyFormat(int format)**
Description		Sets the format of barcode hardware key
Parameter		**Format** - 0 : PTT/SCAN - 1 : SCAN/PTT - 2 : PTT / PTT - 3 : SCAN / SCAN
Return	**Reader**	**Success** SDConsts.BCResult.***SUCCESS*** = 0 **Argument Error** SDConsts.SDResult.***ARGUMENT_ERROR*** = -3 **Other Errors** SDConsts.BCResult.***OTHER_ERROR*** = -1 *** Can receive other error constant of "BCResult" class.**
	BTReader	**Success** SDConsts.BCResult.***SUCCESS*** = 0 **Argument Error** SDConsts.BCResult.***ARGUMENT_ERROR*** = -3 **Not Supported Errors** SDConsts.BCResult.***NOT_SUPPORTED_API*** = -36 **Other Errors** SDConsts.BCResult.***OTHER_ERROR*** = -1 *** Can receive other error constant of "BCResult" class.**
Remark		**※ Reference (3.2.BCKeyFormat)** **Device has two barcode keys on both side.** **In the case of A/B format, left side key means A and right means B.** **If the key is set to "SCAN", it can be use as barcode trigger key.** **Support only Bluebird Android Device with barcode** **(Not Supported on other devices)**

		BC_GetBarcodeKeyFormat
Declare		**public int BC_GetBarcodeKeyFormat()**
Description		Gets the format of barcode hardware key
Parameter		void
Return	**Reader**	**Success :** Value of the barcode key format **(0 : PTT/SCAN, 1 : SCAN/PTT, 2 : PTT/PTT, 3 : SCAN/SCAN)** **Not Supported Errors** SDConsts.BCResult.***NOT_SUPPORTED_API*** = -36 *** Can receive other error constant of "BCResult" class.**

	BTReader	**Success :** Value of the barcode key format **(0 : PTT/SCAN, 1 : SCAN/PTT, 2 : PTT/PTT, 3 : SCAN/SCAN)** **Not Supported Errors** SDConsts.BCResult.***NOT_SUPPORTED_API*** = -36 *** Can receive other error constant of “BCResult” class.**
Remark		**※ Reference (3.2.BCKeyFormat)** **Support only Bluebird Android Device with barcode** **(Not Supported on other devices)**

		BC_SetBarcodeTriggerMode
Declare		**public int BC_SetBarcodeTriggerMode(int BCBarcodeTriggerMode)**
Description		Sets the TriggerMode of BC barcode
Parameter		BCBarcodeTriggerMode - LEVEL : 0 - PULSE : 1 - EDGE : 2 - AUTOSTAND : 3
Return	**Reader**	**Success :** SDConsts.BCResult.***SUCCESS*** = 0 **Argument Error** SDConsts.BCResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.BCResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.BCResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.BCResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Other Errors** SDConsts.BCResult.***OTHER_ERROR*** = -1 *** Can receive other error constant of “BCResult” class**
	BTReader	**Success** SDConsts.BCResult.***SUCCESS*** = 0 **Argument Error** SDConsts.BCResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** SDConsts.BCResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Block State Error** SDConsts.BCResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.BCResult.***READER_OR_COM_INTERFACE_STATUS_ERROR*** = -7 **Command State Error** SDConsts.BCResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Errors** SDConsts.BCResult.***ERROR_HOTSWAP_STATE*** = -37 **Other Errors** SDConsts.BCResult.***OTHER_ERROR*** = -1 *** Can receive other error constant of “BCResult” class.**
Remark		**Support only Bluebird Android Device with barcode** **(Not Supported on other devices)**

BC_GetBarcodeTriggerMode		
Declare		**public int BC_GetBarcodeTriggerMode()**
Description		Gets the TriggerMode of BC barcode
Parameter		void
Return	**Reader**	**Success : :** Value of the barcode trigger mode (LEVEL(0) ~ AUTOSTAND(3)) **Serial Error** SDConsts.BCBarcodeTriggerMode.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.BCBarcodeTriggerMode.***READER_OR_COM_INTERFACE_STATUS_ERROR*** = -7 **Command State Error** SDConsts.BCBarcodeTriggerMode.***OTHER_CMD_RUNNING_ERROR*** = -4 *** Can receive other error constant of "BCBarcodeTriggerMode" class**
	BTReader	**Success : :** Value of the barcode trigger mode (LEVEL(0) ~ AUTOSTAND(3)) **Enabled Error** SDConsts.BCBarcodeTriggerMode.***BLUETOOTH_NOT_ENABLED*** = -15 **Block State Error** SDConsts.BCBarcodeTriggerMode.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.BCBarcodeTriggerMode.***READER_OR_COM_INTERFACE_STATUS_ERROR*** = -7 **Command State Error** SDConsts.BCBarcodeTriggerMode.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Errors** SDConsts.BCBarcodeTriggerMode.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "BCBarcodeTriggerMode" class**
Remark		**※ Reference (3.2.BCBarcodeTriggerMode)** **Support only Bluebird Android Device with barcode** **(Not Supported on other devices)**

BC_SetBarcodeMultiScan		
Declare		**public int BC_SetBarcodeMultiScan(int BCMultiScanState)**
Description		Enable / Disable the MultiScan Mode of BC barcode
Parameter		BCMultiScanState - DISABLE : 0 - ENABLE : 1
Return	**Reader**	**Success :** SDConsts.BCResult.***SUCCESS*** = 0 **Argument Error** SDConsts.BCResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.BCResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.BCResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.BCResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Other Errors** SDConsts.BCResult.***OTHER_ERROR*** = -1 *** Can receive other error constant of "BCResult" class**
	BTReader	**Success :** SDConsts.BCResult.***SUCCESS*** = 0

		Argument Error SDConsts.BCResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** SDConsts.BCResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Block State Error** SDConsts.BCResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.BCResult.***READER_OR_COM_INTERFACE_STATUS_ERROR*** = -7 **Command State Error** SDConsts.BCResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Errors** SDConsts.BCResult.***ERROR_HOTSWAP_STATE*** = -37 **Other Errors** SDConsts.BCResult.***OTHER_ERROR*** = -1 *** Can receive other error constant of "BCResult" class**
Remark		**Support only Bluebird Android Device with barcode** **(Not Supported on other devices)**

		BC_GetBarcodeMultiScanState
Declare		**public int BC_GetBarcodeMultiScanState()**
Description		Gets the MultiScan Mode state of BC barcode
Parameter		void
Return	**Reader**	**Success : :** Value of the barcode multi scan mode state (DISABLE(0) ~ ENABLE(1)) **Serial Error** SDConsts.BCBarcodeTriggerMode.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.BCBarcodeTriggerMode.***READER_OR_COM_INTERFACE_STATUS_ERROR*** = -7 **Command State Error** SDConsts.BCBarcodeTriggerMode.***OTHER_CMD_RUNNING_ERROR*** = -4 *** Can receive other error constant of "BCMultiScanState" class**
	BTReader	**Success : :** Value of the barcode multi scan mode state (DISABLE(0) ~ ENABLE(1)) **Enabled Error** SDConsts.BCMultiScanState.***BLUETOOTH_NOT_ENABLED*** = -15 **Block State Error** SDConsts.BCMultiScanState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.BCMultiScanState.***READER_OR_COM_INTERFACE_STATUS_ERROR*** = -7 **Command State Error** SDConsts.BCMultiScanState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Errors** SDConsts.BCMultiScanState.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "BCMultiScanState" class**
Remark		**※ Reference (3.2.BCMultiScanState)** **Support only Bluebird Android Device with barcode** **(Not Supported on other devices)**

		BC_SetBarcodeMultiScanNumber
Declare		**public int BC_SetBarcodeMultiScanNumber(int BCBarcodeMultiNumber)**
Description		Sets the MultiScan Number of BC barcode

Parameter		BCBarcodeMultiNumber - MIN (1) ~ MAX (10)
Return	**Reader**	**Success :** SDConsts.BCResult.***SUCCESS*** = 0 **Argument Error** SDConsts.BCResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.BCResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.BCResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.BCResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Other Errors** SDConsts.BCResult.***OTHER_ERROR*** = -1 *** Can receive other error constant of "BCResult" class**
	BTReader	**Success :** SDConsts.BCResult.***SUCCESS*** = 0 **Argument Error** SDConsts.BCResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** SDConsts.BCResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Block State Error** SDConsts.BCResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.BCResult.***READER_OR_COM_INTERFACE_STATUS_ERROR*** = -7 **Command State Error** SDConsts.BCResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Errors** SDConsts.BCResult.***ERROR_HOTSWAP_STATE*** = -37 **Other Errors** SDConsts.BCResult.***OTHER_ERROR*** = -1 *** Can receive other error constant of "BCResult" class**
Remark		**Support only Bluebird Android Device with barcode** **(Not Supported on other devices)**

BC_GetBarcodeMultiScanNumber

Declare		**public int BC_GetBarcodeMultiScanNumber()**
Description		Gets the MultiScan Number of BC barcode
Parameter		void
Return	**Reader**	**Success : :** Value of the barcode multi scan number (1 ~ 10) **Serial Error** SDConsts.BCBarcodeTriggerMode.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.BCBarcodeTriggerMode.***READER_OR_COM_INTERFACE_STATUS_ERROR*** = -7 **Command State Error** SDConsts.BCBarcodeTriggerMode.***OTHER_CMD_RUNNING_ERROR*** = -4 *** Can receive other error constant of "BCMultiScanNumber" class**
	BTReader	**Success : :** Value of the barcode multi scan number (1 ~ 10) **Enabled Error** SDConsts.BCMultiScanNumber.***BLUETOOTH_NOT_ENABLED*** = -15 **Block State Error** SDConsts.BCMultiScanNumber.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.BCMultiScanNumber.***READER_OR_COM_INTERFACE_STATUS_ERROR*** = -7 **Command State Error** SDConsts.BCMultiScanNumber.***OTHER_CMD_RUNNING_ERROR*** = -4

		Hotswap Errors SDConsts.BCMultiScanNumber.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "BCMultiScanNumber" class**
Remark		**※ Reference (3.2.BCMultiScanNumber)** **Support only Bluebird Android Device with barcode** **(Not Supported on other devices)**

		BC_SetBarcodeMultiScanType
Declare		**public int BC_SetBarcodeMultiScanType(int BCMultiScanType)**
Description		Enable / Disable the MultiScan Type of BC barcode
Parameter		BCMultiScanType - DISABLE : 0 - ENABLE : 1 * If Enable, you can read only multi number barcode. If Disable, you can read multi and single number barcode.
Return	**Reader**	**Success :** SDConsts.BCResult.***SUCCESS*** = 0 **Argument Error** SDConsts.BCResult.***ARGUMENT_ERROR*** = -3 **Serial Error** SDConsts.BCResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.BCResult.***READER_OR_SERIAL_STATUS_ERROR*** = -7 **Command State Error** SDConsts.BCResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Other Errors** SDConsts.BCResult.***OTHER_ERROR*** = -1 *** Can receive other error constant of "BCResult" class**
	BTReader	**Success :** SDConsts.BCResult.***SUCCESS*** = 0 **Argument Error** SDConsts.BCResult.***ARGUMENT_ERROR*** = -3 **Enabled Error** SDConsts.BCResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Block State Error** SDConsts.BCResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.BCResult.***READER_OR_COM_INTERFACE_STATUS_ERROR*** = -7 **Command State Error** SDConsts.BCResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Errors** SDConsts.BCResult.***ERROR_HOTSWAP_STATE*** = -37 **Other Errors** SDConsts.BCResult.***OTHER_ERROR*** = -1 *** Can receive other error constant of "BCResult" class**
Remark		**Support only Bluebird Android Device with barcode** **(Not Supported on other devices)**

		BC_GetBarcodeMultiScanType
Declare		**public int BC_GetBarcodeMultiScanType()**

Description		Gets the MultiScan Type state of BC barcode
Parameter		void
Return	**Reader**	**Success : :** Value of the barcode multi scan type(0 ~ 1) **Serial Error** SDConsts.BCBarcodeTriggerMode.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.BCBarcodeTriggerMode.***READER_OR_COM_INTERFACE_STATUS_ERROR*** = -7 **Command State Error** SDConsts.BCBarcodeTriggerMode.***OTHER_CMD_RUNNING_ERROR*** = -4 *** Can receive other error constant of "BCMultiScanState" class**
	BTReader	**Success : :** Value of the barcode multi scan type(0 ~ 1) **Enabled Error** SDConsts.BCMultiScanState.***BLUETOOTH_NOT_ENABLED*** = -15 **Block State Error** SDConsts.BCMultiScanState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.BCMultiScanState.***READER_OR_COM_INTERFACE_STATUS_ERROR*** = -7 **Command State Error** SDConsts.BCMultiScanState.***OTHER_CMD_RUNNING_ERROR*** = -4 **Hotswap Errors** SDConsts.BCMultiScanState.***ERROR_HOTSWAP_STATE*** = -37 *** Can receive other error constant of "BCMultiScanState" class**
Remark		**※ Reference (3.2.BCMultiScanNumber)** **Support only Bluebird Android Device with barcode** **(Not Supported on other devices)**

		BC_GetSupportedDevicesInfo
Declare		**public int BC_GetSupportedDevicesInfo()**
Description		Check whether the device supports barcode/camera(Only Bluebird devices that support the feature are available)
Parameter		void
Return	**Reader**	**Success :** BC barcode supporting value(0~3) -Not support Barcode/Camera = 0 -Support only Camera = 1 -Support only Barcode = 2 -Support Camera and Barcode = 3 **Serial Error** SDConsts.BCBarcodeTriggerMode.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts.BCBarcodeTriggerMode.***READER_OR_COM_INTERFACE_STATUS_ERROR*** = -7 **Command State Error** SDConsts.BCBarcodeTriggerMode.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SDResult.***NOT_SUPPORTED_API*** = -36 **Not Active Barcode Error** : Constants.BCResult.***BARCODE_NOT_ACTIVE*** = -35 **Other Errors** : SDConsts.BCResult.***OTHER_ERROR*** = -1 *** Can receive other error constant of "BCMultiScanState" class**

	BTReader	**Success :** BC barcode supporting value(0~3) -Not support Barcode/Camera = 0 -Support only Camera = 1 -Support only Barcode = 2 -Support Camera and Barcode = 3 **Enabled Error** SDConsts. SDResult.***BLUETOOTH_NOT_ENABLED*** = -15 **Block State Error** SDConsts. SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Condition Error** SDConsts. SDResult.***READER_OR_COM_INTERFACE_STATUS_ERROR*** = -7 **Command State Error** SDConsts. SDResult.***OTHER_CMD_RUNNING_ERROR*** = -4 **Not Supported Error** : SDConsts.SDResult.***NOT_SUPPORTED_API*** = -36 **Not Active Barcode Error** : Constants.BCResult.***BARCODE_NOT_ACTIVE*** = -35 **Other Errors** : SDConsts.BCResult.***OTHER_ERROR*** = -1 *** Can receive other error constant of “BCMultiScanState” class**
Remark		**※ Reference (3.2.BCMultiScanNumber)** **Support only Bluebird Android Device with barcode** **(Not Supported on other devices)**

■ BT APIs

BT_Enable	
Declare	**public boolean BT_Enable()**
Description	Enable Bluetooth
Parameter	Void
BTReader Return	**Success :** True(Enable Bluetooth / Already enabled) **Fail** : False
Remark	**※ This API is only for Bluetooth interface(BTReader)** **[Requires permission]** **- android.Manifest.permission.BLUETOOTH** **- android.Manifest.permission.BLUETOOTH_ADMIN**

BT_Disable	
Declare	**public boolean BT_Disable()**
Description	Disable Bluetooth
Parameter	Void
BTReader Return	**Success :** True(Disable Bluetooth / Already disabled) **Fail** : False
Remark	**※ This API is only for Bluetooth interface(BTReader)** **[Requires permission]** **- android.Manifest.permission.BLUETOOTH** **- android.Manifest.permission.BLUETOOTH_ADMIN**

BT_IsEnabled	
Declare	**public boolean BT_IsEnabled()**
Description	Check Bluetooth enable state
Parameter	Void
BTReader Return	**Success :** True(Enabled) **Fail** : False(Not enabled)
Remark	**※ This API is only for Bluetooth interface(BTReader)** **[Requires permission]** **- android.Manifest.permission.BLUETOOTH**

BT_GetPairedDevices

Declare	**public Set<BluetoothDevice> BT_GetPairedDevices()**
Description	Gets Paired SLED Device list
Parameter	Void
BTReader Return	**Success :** Get paired SLED device list **Fail** : Null(Not enable state)
Remark	**※ This API is only for Bluetooth interface(BTReader)** **[Requires permission]** - **android.Manifest.permission.BLUETOOTH**

BT_StartScan

Declare	**public boolean BT_StartScan()**
Description	Start Bluetooth scan, but it will be not working in Bluetooth connected state.
Parameter	Void
BTReader Return	**Success :** True(Start Bluetooth scan) **Fail** : False(Can't start Bluetooth scan)
Remark	**※ This API is only for Bluetooth interface(BTReader)** **[Requires permission]** - **android.Manifest.permission.BLUETOOTH** - **android.Manifest.permission.BLUETOOTH_ADMIN**

BT_StopScan

Declare	**public boolean BT_StopScan()**
Description	Stop Bluetooth scan
Parameter	Void
BTReader Return	**Success :** True(Stop Bluetooth scan) **Fail** : False(Can't stop Bluetooth scan)
Remark	**※ This API is only for Bluetooth interface(BTReader)** **[Requires permission]** - **android.Manifest.permission.BLUETOOTH** - **android.Manifest.permission.BLUETOOTH_ADMIN**

BT_Connect

Declare	**public int BT_Connect(String address)**

Description	Connect bluetooth device with bt address information
Parameter	address - Bluetooth address
BTReader Return	**Success :** SDConsts.BTResult.***SUCCESS*** = 0 **Connect Error** : ~~SDConsts.BTResult.*ALREADY_CONNECTING* = -18~~ SDConsts.BTResult.***ALREADY_CONNECTED*** = -10 **State Error** : SDConsts.BTResult.***BT_NOT_ENABLE_STATE*** = -40 *** Can receive other error constant of "BTResult" class.**
Remark	**※ This API is only for Bluetooth interface(BTReader)** **[Requires permission]** - **android.Manifest.permission.BLUETOOTH** - **android.Manifest.permission.BLUETOOTH_ADMIN** **※ Bluetooth can be connected through the existing BT_Connect(String address) API, but it may take a little longer than BT_Connect(String address, String deviceType) API.**

	BT_Connect
Declare	**public int BT_Connect(String address, String deviceType)**
Description	Connect bluetooth device with bt address & type information
Parameter	address - Bluetooth address **deviceType** (SDConsts.BTDeviceType) : Bluetooth device type (If you don't know the type, you can use null) - TYPE_1 = "01"; - TYPE_2 = "02"; - TYPE_3 = "03";
BTReader Return	**Success :** SDConsts.BTResult.***SUCCESS*** = 0 **Connect Error** : ~~SDConsts.BTResult.*ALREADY_CONNECTING* = -18~~ SDConsts.BTResult.***ALREADY_CONNECTED*** = -10 **State Error** : SDConsts.BTResult.***BT_NOT_ENABLE_STATE*** = -40 *** Can receive other error constant of "BTResult" class.**
Remark	**※ This API is only for Bluetooth interface(BTReader)** **[Requires permission]** - **android.Manifest.permission.BLUETOOTH** - **android.Manifest.permission.BLUETOOTH_ADMIN** **※ It is mainly used when connecting using NFC or QR Code.** **※ Bluetooth can be connected through the existing BT_Connect(String address) API, but it may take a little longer than BT_Connect(String address, String deviceType) API.**

BT_Disconnect	
Declare	**public int BT_Disconnect()**
Description	Disconnect Bluetooth device
Parameter	void
BTReader Return	**Success :** SDConsts.BTResult.***SUCCESS*** = 0 **Connect Error** : SDConsts.BTResult.***ALREADY_DISCONNECTED*** = -9 **State Error** : SDConsts.BTResult.***BT_NOT_ENABLE_STATE*** = -40 *** Can receive other error constant of "BTResult" class.**
Remark	**※ This API is only for Bluetooth interface(BTReader)** **[Requires permission]** - **android.Manifest.permission.BLUETOOTH** - **android.Manifest.permission.BLUETOOTH_ADMIN**

BT_GetConnectState	
Declare	**public int BT_GetConnectState()**
Description	Gets connect state of Bluetooth device
Parameter	void
BTReader Return	SDConsts.BTConnectState.***NONE*** = 0 SDConsts.BTConnectState.***CONNECTING*** = -1 SDConsts.BTConnectState.***CONNECTED*** = 2 *** Can receive other error constant of "BTConnectState" class.**
Remark	**※ This API is only for Bluetooth interface(BTReader)** **[Requires permission]** - **android.Manifest.permission.BLUETOOTH**

BT_UnpairDevice	
Declare	**public boolean BT_UnpairDevice(String address)**
Description	Unpair paired Bluetooth device
Parameter	address - Bluetooth address
BTReader Return	**Success :** True(Unpair device) **Fail** : False
Remark	**※ This API is only for Bluetooth interface(BTReader)**

	[Requires permission] - **android.Manifest.permission.BLUETOOTH** - **android.Manifest.permission.BLUETOOTH_ADMIN**

BT_UnpairAllDevices	
Declare	**public boolean BT_UnpairAllDevices()**
Description	Unpair All paired Bluetooth device
Parameter	void
BTReader Return	**Success :** True(Unpair all devices) **Fail** : False
Remark	**※ This API is only for Bluetooth interface(BTReader)** **[Requires permission]** - **android.Manifest.permission.BLUETOOTH** - **android.Manifest.permission.BLUETOOTH_ADMIN**

BT_GetConnectedDeviceName	
Declare	**public String BT_GetConnectedDeviceName()**
Description	Gets connected device name
Parameter	void
BTReader Return	**Success :** Connected device name **Fail** : Null
Remark	**※ This API is only for Bluetooth interface(BTReader)** **[Requires permission]** - **android.Manifest.permission.BLUETOOTH**

BT_GetConnectedDeviceAddr	
Declare	**public String BT_GetConnectedDeviceAddr()**
Description	Gets connected device address
Parameter	void
BTReader Return	**Success :** Connected device address **Fail** : Null
Remark	**※ This API is only for Bluetooth interface(BTReader)** **[Requires permission]** - **android.Manifest.permission.BLUETOOTH**

4. Special note

1) Document Conventions

- The latest changes are in blue letters on a yellow background.
- Highlights are as red letters.
- Deleted changes are indicated by gray letters and middle lines
- Deprecated APIs are in white letters on a red background.
- Only for Serial(Reader)/Bluetooth(BTReader) interface's APIs are in white letters on a blue background
- Requires permission infomations are as green letters.