RS9113 WiSeConnect™
Bluetooth Classic Software Programming Reference Manual

Version 1.5.0

March 2016

Redpine Signals, Inc.

2107 N. First Street, #680

San Jose, CA 95131.

Tel: (408) 748-3385

Disclaimer:

The information in this document pertains to information related to Redpine Signals, Inc. products. This information is provided as a service to our customers, and may be used for information purposes only.

Redpine assumes no liabilities or responsibilities for errors or omissions in this document. This document may be changed at any time at Redpine's sole discretion without any prior notice to anyone. Redpine is not committed to updating this document in the future.

About this Document

This document describes the commands to operate the RS9113-WiSeConnect Module Family for Bluetooth. Bluetooth stack is used for Host layers and the commands describe various profiles supported by Bluetooth stack. This document should be used by the developer to write software on Host MCU to control and operate the module.

Table of Contents

Table of Figures

Table of Tables

1 Overview

This document describes the commands to operate RS9113-WiSeConnect Module Family in Bluetooth. The parameters in the commands and their valid values with the expected responses from the modules are also described. The document should be used by the developer to write software on the Host MCU to control and operate the module.

2 Bluetooth Software Programming

The following sections describe Bluetooth software architecture and commands to operate and configure the RS9113 modules in Bluetooth.

2.1 Bluetooth Software Architecture

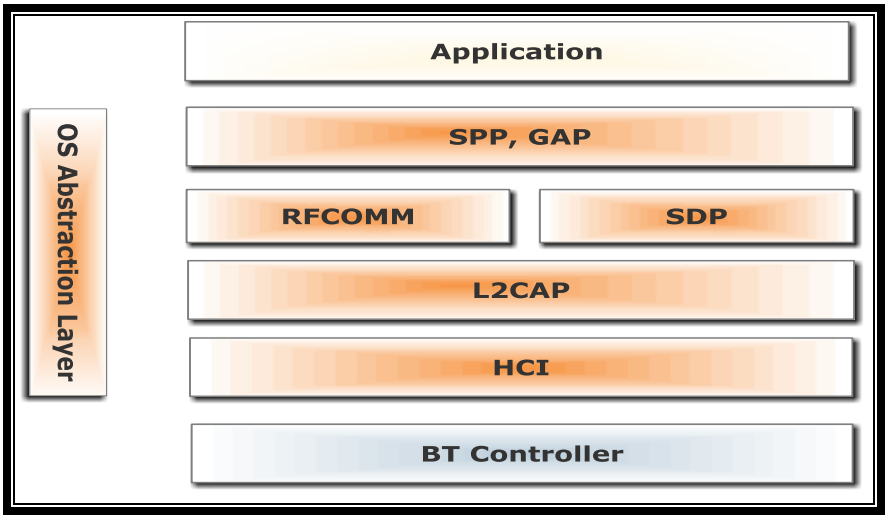

Figure 1: Bluetooth Software Architecture Application

The application layer launches the Bluetooth stack and uses the commands to access various profiles on the remote Bluetooth devices over the network.

2.1.2 Profiles

There are number of Bluetooth profiles defined in the Bluetooth specification. We currently supports profiles including Serial Port Profile (SPP). We provide framework to develop new profiles very easily. We will continue to add new profiles.

2.1.3 Bluetooth Core

The Bluetooth core contains the following higher layers of the stack.

RFCOMM

SDP

L2CAP

HCI Generic Driver

HCI BUS Driver

RFCOMM is a transport protocol based on L2CAP. It emulates RS-232 serial ports. The RFCOMM protocol supports up to 60 simultaneous connections between two BT devices. RFCOMM provides data stream interface for higher level applications and profiles.

SDP (Service Discovery Protocol) provides a means for applications to discover which services are available and to determine the characteristics of those available services. SDP uses an existing L2CAP connection.

Further connection to Bluetooth devices can be established using information obtained via SDP.

L2CAP (Logical Link Control and Adaptation Protocol) provides connection-oriented and connectionless data services to upper layer protocols with data packet size up to 64 KB in length. L2CAP performs the segmentation and reassemble of I/O packets from the baseband controller.

HCI Generic Driver – This driver implements the HCI Interface standardized by Bluetooth SIG. It establishes the communication between the Stack and the HCI Firmware in the Bluetooth hardware. It communicates with the Bluetooth controller hardware via the HCI Bus driver.

HCI Transport Layer Driver – The Bluetooth controllers are connected to the host using interface like UART, USB, SDIO, SPI, USB-CDC etc. The HCI Transport Layer Driver provides hardware abstraction to the rest of the Bluetooth stack software. This driver makes it possible to use Bluetooth stack with different hardware interfaces.

2.1.4 OS Abstraction Layer

This layer abstracts RTOS services (semaphores, mutexes and critical sections) that are used by the whole stack and the applications. The stack, which is designed in an RTOS-independent manner, can be used with any RTOS by porting this layer. It is also possible to use the Bluetooth stack standalone without RTOS.

2.2 Bluetooth Command Format

This section explains the general command format. The commands should be sent to the Module in the specified format.

The commands are sent to the module and the responses are read from the module using frame write/frame read (as mentioned in the preceeding sections). These commands are called as command frames.

The format of the command frames are divided into two parts:

1. Frame descriptor
2. Frame Body(Frame body is often called as Payload)

Frame Descriptor (16 bytes)	**Frame Body (multiples of 4 bytes)**

Command frame format is shown below. This description is for a Little Endian System.

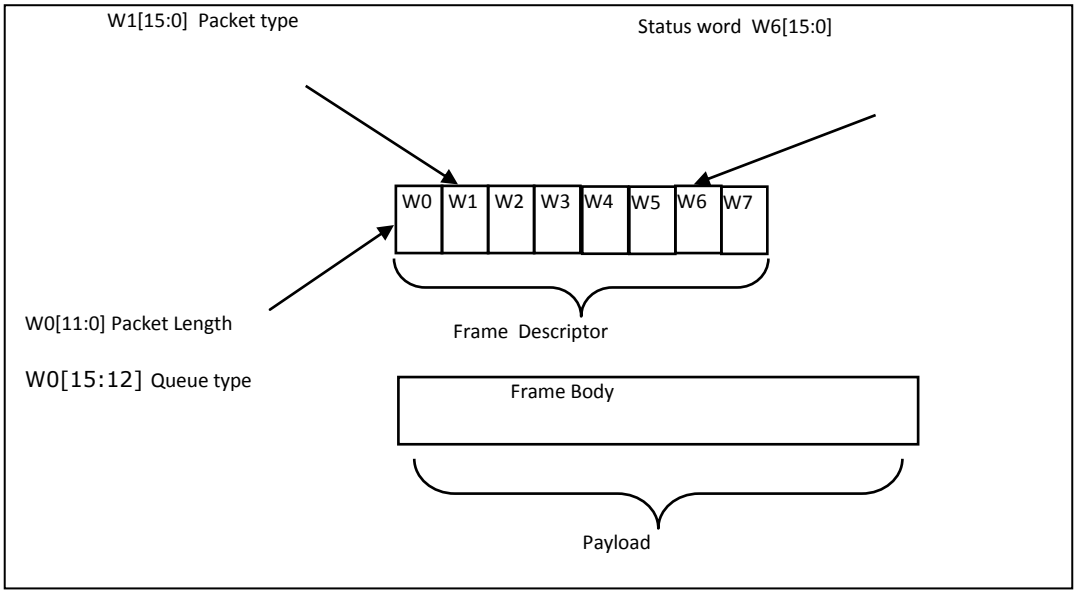

Figure 2: Command frame format

The following table provides the general description of the frame descriptor.

Word	Frame Descriptor
Word0 W0[15:0]	Bits [11:0] – Length of the frame Bits [15:12] – 2(indicates Bluetooth packet).
Word1 W1[15:0]	Bits [15:0] - Packet type
Word2 W2[15:0]	Reserved
Word3 W3[15:0]	Reserved
Word4 W4[15:0]	Reserved
Word5 W5 [15:0]	Reserved
Word6 W6 [15:0]	1. (0x0000) when sent from host to module. 2. When sent from module to host (as response frame), it contains the status.

Word	Frame Descriptor
Word7 W7 [15:0]	Reserved

Table 1 Frame Descriptor

Three types of frames will get exchanged between the module and the host.

1. Request/Command frames - These are sent from Host to Module. Each Request/ Command has an asscociated response with it.
2. Response frames – These are sent from Module to Host. These are given in response to the previous Request/Command from the Host. Each command has a single reponse.
3. Event frames – These are sent from Module to Host. These are given when
   a) There are multiple reponses for a particular Request/ Command frame
   b) There is Asynchonous message to be sent to host.

The following are the types of frame requests and responses and the corresponding codes. The commands are different for both Classic and LE modes. The below table lists the Command, Response and Event frames in Classic mode.

In both the modes, the corresponding code is to be filled in W1 [15:0] mentioned in the table above.

Command	Command ID
Set Local Name	0x0001
Query Local Name	0x0002
Set Local COD	0x0003
Query Local COD	0x0004
Query RSSI	0x0005
Query Link Quality	0x0006
Query Local BD Address	0x0007
Set Profile Mode	0x0008
Set Device Discover Mode	0x0009
Get Device Discover Mode	0x000A

Set Connection mode	0x000B
Get Connection mode	0x000C
Set Pair mode	0x000D
Get Pair mode	0x000E
Remote Name Request	0x000F
Remote Name Request Cancel	0x0010
Inquiry	0x0011
Inquiry Cancel	0x0012
Bond or Create Connection	0x0013
Bond Cancel or Create Connection Cancel	0x0014
Unbond or Disconnect	0x0015
Set Pin Type	0x0016
Get Pin Type	0x0017
User Confirmation	0x0018
Passkey Reply	0x0019
Pincode Reply	0x001A
Get Local Device Role	0x001B
Set Local Device Role	0x001C
Get Service List	0X001D
Search Service	0X001E
SPP connect	0X001F
SPP Disconnect	0X0020
SPP Transfer	0X0021
Initialize BLE module	0x008D
Deinitialize BLE module	0x008E
Antenna Select	0x008F
Linkkey Reply	0x0091
PER Transmit	0x0098
PER Receive	0x0099
PER Stats	0x009A
PER CW mode	0x009B
Sniff Mode	0x009D

Sniff Exit	0x009E
Sniff Subrating	0x009F

Table 2 Command IDs in BT Classic mode

Response	Response ID
Card ready	0x0505
Set Local Name	0x0001
Query Local Name	0x0002
Set Local COD	0x0003
Query Local COD	0x0004
Query RSSI	0x0005
Query Link Quality	0x0006
Query Local BD Address	0x0007
Set Profile Mode	0x0008
Set Device Discover Mode	0x0009
Get Device Discover Mode	0x000A
Set Connection mode	0x000B
Get Connection mode	0x000C
Set Pair mode	0x000D
Get Pair mode	0x000E
Remote Name Request	0x000F
Remote Name Request Cancel	0x0010
Inquiry	0x0011
Inquiry Cancel	0x0012
Bond or Create Connection	0x0013
Bond Cancel or Create Connection Cancel	0x0014
Unbond or Disconnect	0x0015
Set Pin Type	0x0016
Get Pin Type	0x0017
User Confirmation	0x0018

Passkey reply	0x0019
Pincode Reply	0x001A
Get Local Device Role	0x001B
Set Local Device Role	0x001C
Get Service List	0X001D
Search Service	0X001E
SPP connect	0X001F
SPP Disconnect	0X0020
SPP Transfer	0X0021
BT Classic init	0x008D
BT Classic deint	0x008E
Antenna Select	0x008F
Linkkey Reply	0x0091
PER Transmit	0x0098
PER Recieve	0x0099
PER Stats	0x009A
PER CW mode	0x009B
SSP Mode	0x00A0
Sniff Mode	0x009D
Sniff Exit	0x009E
Sniff Subrating	0x009F

Table 3 Response IDs in BT Classic mode

Event	Event ID
Role change status	0x1000
Unbond or Disconnect	0x1001
Bond Response	0x1002
Inquiry response	0x1003
Remote Device Name	0x1004
Remote Name Request cancelled	0x1005
Disconnected	0x1006
User confirmation request	0x1007

User passkey display	0x1008
User pincode request	0x1009
User passkey request	0x100A
Inquiry complete	0x100B
Auth complete	0x100C
User linkkey request	0x100D
User linkkey save	0x100E
SPP Receive	0x1100
SPP connected	0x1101
SPP Disconnected	0x1102
Mode Changed	0x1010
Sniff Subrating Changed	0x1011

Table 4 Event IDs in BT Classic mode

2.3 Bluetooth Classic commands

The following sections will explain various RS9113-WiSeConnect Bluetooth Classic commands, their structures, the parameters they take and their responses. For API prototypes of these commands, please refer to the API Library Section

Note: All BT/BLE AT command are case sensitive and lower case.

NOTE: A new command has to be called only after getting the response for the previous command

2.3.1 Generic commands

2.3.1.1 Set Local name

Description: This is used to set name to the local device.

Binary Payload Structure:

```
typedef struct rsi_bt_cmd_set_local_name {
        UINT08  NameLength;
        INT08   Name[50];
    } RSI_BT_CMD_SET_LOCAL_NAME;
```

AT command format:

at+rsibt_setlocalname=<NameLength>,<Name>\r\n

Parameters:

NameLength – Length of the name of the local device.

Name – Name of the local device.

Response Payload:

There is no response payload for this command.

AT command Ex: at+rsibt_setlocalname=8,redpines\r\n

Response: OK\r\n

2.3.1.2 Query Local name

Description: This is used to query the name of the local device.

Binary Payload Structure:

No Payload required.

AT command format:

at+rsibt_getlocalname?\r\n

Response Payload:

typedef struct rsi_bt_resp_query_local_name {

UINT08 NameLength;

INT08 Name[50];

} RSI_BT_RESP_QUERY_LOCAL_NAME;

Result Code	Description
OK <name_length>,<local_device_name>	Command Success.
ERROR <Error_code>	Command Fail.

Response Parameters:

NameLength – Length of the name of the local device.

Name – Name of the local device.

AT command Ex: at+rsibt_getlocalname?\r\n

Response: OK 8,redpines\r\n

2.3.1.3 Set Local COD

Description: This is used to indicate the capabilities of the local device to other devices. It is a parameter received during the device discovery procedure on the BR/EDR physical transport, indicating the type of device. The Class of Device parameter is only used on BR/EDR and BR/EDR/LE devices using the BR/EDR physical transport.

Binary Payload Structure:

typedef struct rsi_bt_cmd_set_local_cod {

UINT32 LocalCOD;

} RSI_BT_CMD_SET_LOCAL_COD;

AT command format:

at+rsibt_setlocalcod=<local_device_class>\r\n

Parameters:

Local COD – Class of the Device of the local device

Response Payload:

There is no response payload for this command.

AT command Ex: at+rsibt_setlocalcod=7A020C\r\n

Response: OK\r\n

2.3.1.4 Query Local COD

Description: This is used to query Class of Device of the local device.

Binary Payload Structure:

No Payload required.

AT command format:

at+rsibt_getlocalcod?\r\n

Response Payload:

```
typedef struct rsi_bt_resp_query_local_cod {
   UINT32 LocalCOD;
} RSI_BT_RESP_QUERY_LOCAL_COD;
```

Result Code	Description
OK <local_device_class>	Command Success.
ERROR <Error_code>	Command Fail.

Response Parameters:

LocalCOD – Class of the Device of the local device

AT command Ex: at+rsibt_getlocalcod?\r\n

Response: OK 7A020C\r\n

2.3.1.5 Query RSSI

Description: This is used to query RSSI of the connected remote BT Device.

Binary Payload Structure:

```
typedef struct rsi_bt_cmd_query_rssi {
        UINT08 BDAddress[6];
} RSI_BT_CMD_QUERY_RSSI;
```

AT command format:

at+rsibt_getrssi=<BDAddress>?\r\n

Parameters:

BDAddress – BD Address of the connected remote device.

Response Payload:

typedef struct rsi_bt_resp_query_rssi {

UINT08 RSSI;

} RSI_BT_RESP_QUERY_RSSI;

Result Code	Description
OK <rssi value>	Command Success.
ERROR <Error_code>	Command Fail.

Response parameters:

RSSI – RSSI value of the connected remote device.

AT command Ex: at+rsibt_getrssi=AA-BB-CC-DD-EE-FF?\r\n

Response: OK 230\r\n

2.3.1.6 Query Link Quality[1]

Description: This is used to query the link quality between the local device and the connected remote device.

Binary Payload Structure:

typedef struct rsi_bt_cmd_query_link_quality {

UINT08 BDAddress[6];

} RSI_BT_CMD_QUERY_LINK_QUALITY;

AT command format:

at+rsibt_getlinkqlty=<BDAddress>?\r\n

Parameters:

BDAddress – BD Address of the connected remote device

Response Payload:

typedef struct rsi_bt_resp_query_link_quality {

UINT08 LinkQuality;

} RSI_BT_RESP_QUERY_LINK_QUALITY;

[1] This command is not currently supported.

Result Code	Description
OK <link_quality>	Command Success with valid response.
ERROR <Error_code>	Command Fail.

Response parameters:

LinkQuality – Link quality value.

AT command Ex: at+rsibt_getlinkqlty=AA-BB-CC-DD-EE-FF?\r\n

Response: OK 123\r\n

2.3.1.7 Query Local BD Address

Description: This is used to query the BD address of the local device.

Binary Payload Structure:

No Payload required.

AT command format:

at+rsibt_getlocalbdaddr?\r\n

Response Payload:

```
typedef struct rsi_bt_resp_query_local_bd_address {
   UINT08  BDAddress[6];
} RSI_BT_RESP_QUERY_LOCAL_BD_ADDRESS;
```

Result Code	Description
OK <bd_addr>	Command Success with valid response.
ERROR <Error_code>	Command Fail.

Response Parameters:

BDAddress - BD Address of the local device

AT command Ex: at+rsibt_getlocalbdaddr?\r\n

Response: OK AA-BB-CC-DD-EE-FF\r\n

2.3.1.8 Initialize BT module

Description: This is used to initialize the BT module.

Binary Payload Structure:

No Payload required

AT command format:.

at+rsibt_btinit\r\n

Response Payload:

There is no response payload for this command.

AT command Ex: at+rsibt_btinit\r\n

Response: OK\r\n

2.3.1.9 Deinitialize BT module

Description: This is used to deinitialize the BT module.To again initialize the module **Initialize BT module** command is used.

Binary Payload Structure:

No Payload required

AT command format:

at+rsibt_btdeinit\r\n

Payload Structure:

No Payload required.

Response Payload:

There is no response payload for this command.

AT command Ex: at+rsibt_btdeinit\r\n

Response: OK\r\n

2.3.1.10 BT Antenna Select

Description: This is used to select the internal or external antenna of the BT module.

Binary Payload Structure:

```
typedef struct rsi_bt_cmd_antenna_select{
        UINT08 AntennaVal;
} RSI_BT_CMD_ANTENNA_SELECT;
```

AT command format:

at+rsibt_btantennaselect=<antenna_val>\r\n

Parameters:

AntennaVal – To select the internal or external antenna

0 – Internal Antenna.

1 – External Antenna.

Response Payload:

There is no response payload for this command.

AT command Ex: at+rsibt_btantennaselect=1\r\n

Response: OK\r\n

2.3.2 Core commands

2.3.2.1 Set Profile Mode[1]

Description: This is used to initialize the particular profiles in Bluetooth embedded host stack.

Binary Payload Structure:

```
typedef struct rsi_bt_cmd_set_profile_mode{

   UINT08  ProfileMode;

}RSI_BT_CMD_SET_PROFILE_MODE;
```

AT command format:

at+rsibt_setprofilemode=<ProfileMode>\r\n

Parameters:

Profile Mode – Set specific bits to enable the profiles.

Bit No	Description
0	SPP Profile
1	A2DP Profile
2	AVRCP Profile
3	HFP Profile
4	PBAP Profile
5	IAP Profile

[1] Present only SPP profile is supported.

Response Payload:

There is no response payload for this command.

AT command Ex: at+rsibt_setprofilemode=1\r\n

Response: OK\r\n

NOTE: According to profile requirements, need to give the bit numbers. For example if you required spp profile + A2DP Profile then u have to give value 3.

2.3.2.2 Set Device Discovery mode

Description: This is used to set the BT module in any of the three Discovery modes. We have to use time out for only limited discovering.

Binary Payload Structure:

```
typedef struct rsi_bt_cmd_set_discv_mode {
  UINT08  Mode;
  UINT08 Reserved[3];
  INT32  Timeout;
} RSI_BT_CMD_SET_DISCV_MODE;
```

AT command format:

at+rsibt_setdiscvmode=<mode>,<timeout>\r\n

Parameters:

Mode – To enable/disable discovering

- 0 – disable discovering
- 1 – enable discovering
- 2 – limited discovering

TimeOut – time out value in milli seconds.

Note: Better to use below 1 hour(i.e.. >3600000ms).

Response Payload:

There is no response payload for this command.

AT command Ex: at+rsibt_setdiscvmode=2,10000\r\n

Response: OK\r\n

2.3.2.3 Get Device Discovery mode

Description: This is used to get the discovery mode of the BT module, currently the BT module was set.

Binary Payload Structure:

There is no payload for this command.

AT command format:

at+rsibt_getdiscvmode?\r\n

Response Payload:

```
typedef struct rsi_bt_resp_query_discovery_mode {

   UINT08  DiscoveryMode;

} RSI_BT_RESP_QUERY_DISCOVERY_MODE;
```

Result Code	Description
OK <mode>	Command Success with valid response.
ERROR <Error_code>	Command Fail.

Response Parameters:

DiscoveryMode – enabled/disabled discovering

0 – Disabled device discover

1 – Enabled device discover

AT command Ex: at+rsibt_ getdiscvmode?\r\n

Response: OK 1\r\n

2.3.2.4 Set Connectability mode

Description: This is used to set the BT module in one of the two Connectability modes.

Binary Payload Structure:

```
typedef struct rsi_bt_cmd_set_connection_mode {

   UINT08  ConnMode;

}RSI_BT_CMD_SET_CONN_MODE;
```

AT command format:

at+rsibt_setconnmode=<ConnMode>\r\n

Parameters:

ConnMode – To enable/disable connectability

0 – disable connection mode

1 – enable connection mode

Response Payload:

There is no response payload for this command.

AT command Ex: at+rsibt_setconnmode=1\r\n

Response: OK\r\n

2.3.2.5 Get Connectablility mode

Description: This is used to get the connectable mode, currently the BT module was set.

Binary Payload Structure:

There is no payload for this command

AT command format:

at+rsibt_getconnmode?\r\n

Response Payload:

```
typedef struct rsi_bt_resp_query_conn_mode {
   UINT08  ConnMode;
} RSI_BT_RESP_QUERY_CONN_MODE;
```

Result Code	Description
OK <mode>	Command Success with valid response.
ERROR <Error_code>	Command Fail.

Response Parameters:

ConnMode – enabled/disabled connection mode

0 – Disabled connection mode

1 – Enabled connection mode

AT command Ex: at+rsibt getconnmode?\r\n

Response: OK 1\r\n

2.3.2.6 Set Pair mode

Description: This will enable or disable the Pairing mode of the BT module.

Binary Payload Structure:

```
typedef struct rsi_bt_cmd_set_pair_mode {
   UINT08  PairMode;
}RSI_BT_CMD_SET_PAIR_MODE;
```

AT command format:

at+rsibt_setpairmode=<PairMode>\r\n

Parameters:

PairMode – To enable/disable Authentication

0 – disable Authentication mode

1 – enable Authentication mode

Response Payload:

There is no response payload for this command.

AT command Ex: at+rsibt_setpairmode=0\r\n

Response: OK\r\n

2.3.2.7 Get Pair mode

Description: This will retrieve the current pairing mode of the BT module.

Binary Payload Structure:

There is no payload for this command.

AT command format:

at+rsibt_getpairmode?\r\n

Response Payload:

```
typedef struct rsi_bt_resp_query_pair_mode {
   UINT08  PairMode;
} RSI_BT_RESP_QUERY_PAIR_MODE;
```

Result Code	Description
OK <mode>	Command Success with valid response.
ERROR <Error_code>	Command Fail.

Response Parameters:

PairMode – enabled/disabled Authentication mode

0 – Disabled Authentication mode

1 – Enabled Authentication mode

AT command Ex: at+rsibt getpairmode?\r\n

Response: OK 0\r\n

2.3.2.8 Remote Name Request

Description: This is used to know the name of the remote BT device, using its BD address. The response to this command containing the remote BT device name will be sent to the host through **"RMTDEVNAME"** event.

Binary Payload Structure:

```
typedef struct rsi_bt_cmd_remote_name_req {

   UINT08  BDAddress[6];

}RSI_BT_CMD_REMOTE_NAME_REQUEST;
```

AT command format:

at+rsibt_rmtnamereq=<BDAddress>\r\n

Parameters:

BDAddress – remote device BD Address

Response Payload:

There is no response payload for this command.

AT command Ex: at+rsibt_rmtnamereq= AA-BB-CC-DD-EE-FF\r\n

Response: OK\r\n

2.3.2.9 Remote Name Request Cancel

Description: This will cancel the request served by "Remote Name Request" command. The cancellation will be confirmed through "Remote Name Request Cancelled" event.

Binary Payload Structure:

```
typedef struct rsi_bt_cmd_remote_name_req_cancel {

   UINT08  BDAddress[6];
```

}RSI_BT_CMD_REMOTE_NAME_REQUEST_CANCEL;

AT command format:

at+rsibt_rmtnamereqcancel=<BDAddress>\r\n

Parameters:

BDAddress – remote device BD Address

Response Payload:

There is no response payload for this command.

AT command Ex: at+rsibt_rmtnamereqcancel= AA-BB-CC-DD-EE-FF \r\n

Response: OK\r\n

2.3.2.10 Inquiry

Description: This will perform an inquiry scan to find any BT devices in the vicinity. The response is sent using **"INQRESP"** event.

Binary Payload Structure:

```
typedef struct rsi_bt_cmd_inquiry {
   UINT08  InquiryType;
   UINT08  Reserved[3];
   UINT32  Duration;
   UINT08  MaxNbrdev;
   UINT08  Reserved[3];
} RSI_BT_CMD_INQUIRY;
```

AT command format:

at+rsibt_inquiry=<InquiryType>,<Duration>,<MaxNbrdev>\r\n

Parameters:

InquiryType –

0- Standard Inquiry

1 – Inquiry with RSSI

2 – Extended Inquiry

Duration – Extended Time in milliseconds (up to 5000ms)

MaxNbrdev – maximum number of devices to scan (from 1 to 5)

Response Payload:

There is no response payload for this command.

AT command Ex: at+rsibt_inquiry=1,10000,10\r\n

Response: OK\r\n

2.3.2.11 Inquiry Cancel

Description: This will cancel the inquiry scan which was already in the process, served by "**Inquiry**" command.

Binary Payload Structure:

There is no payload Structure for this command.

AT command format:

at+rsibt_inquirycancel\r\n

Response Payload:

There is no response payload for this command.

AT command Ex: at+rsibt_inquirycancel\r\n

Response: OK\r\n

2.3.2.12 Bond or Create Connection

Description: This will creates bonding (connection) between the BT module and the remote BT device based on BD address along with security.

Binary Payload Structure:

```
typedef struct rsi_bt_cmd_bond {
   UINT08  BDAddress[6];
}RSI_BT_CMD_BOND;
```

AT command format:

at+rsibt_bond=<BDAddress>\r\n

Parameters:

BDAddress – remote device BD Address.

Response Payload:

There is no response payload for this command.

AT command Ex: at+rsibt_bond= AA-BB-CC-DD-EE-FF\r\n

Response: OK\r\n

2.3.2.13 Bond Cancel or Create Connection Cancel

Description: This will disconnect the connection between the BT module and the remote BT device, only while the bonding is in progress.

Binary Payload Structure:

```
typedef struct rsi_bt_cmd_bond_cancel {
   UINT08  BDAddress[6];
}RSI_BT_CMD_BOND_CANCEL;
```

AT command format:

at+rsibt_bondcancel=<BDAddress>\r\n

Parameters:

BDAddress – remote device BD Address

Response Payload:

There is no response payload for this command.

AT command Ex: at+rsibt_bondcancel = AA-BB-CC-DD-EE-FF\r\n

Response: OK\r\n

2.3.2.14 UnBond or Disconnect

Description: This un-bonds the device, which was already bonded, based on BD address of the remote BT device.

Binary Payload Structure:

```
typedef struct rsi_bt_cmd_unbond {
   UINT08  BDAddress[6];
}RSI_BT_CMD_UNBOND;
```

AT command format:

at+rsibt_unbond=<BDAddress>\r\n

Parameters:

BDAddress – remote device BD Address

Response Payload:

There is no response payload for this command.

AT command Ex: at+rsibt_unbond= AA-BB-CC-DD-EE-FF\r\n

Response: OK\r\n

2.3.2.15 Set Pin type[1]

Description: This is used to set the PIN code or pass key of the local BT module.

Binary Payload Structure:

```
typedef struct rsi_bt_cmd_set_pin_type {
   UINT08  PINType;
}RSI_BT_CMD_SET_PIN_TYPE;
```

AT command format:

at+rsibt_setpintype=<PINType>\r\n

Parameters:

PINType 0 – variable pin

1 – fixed pin

Response Payload:

There is no response payload for this command.

AT command Ex: at+rsibt_setpintype=1\r\n

Response: OK\r\n

2.3.2.16 Get Pin type[2]

Description: This is used to get the PIN code or pass key of the local BT module.

Binary Payload Structure:

There is no response payload for this command.

AT command format:

at+rsibt_getpintype?\r\n

Response Payload:

```
typedef struct rsi_bt_resp_query_pin_type {
   UINT08  PINType;
} RSI_BT_RESP_QUERY_PIN_TYPE;
```

[1] This command is not currently supported.
[2] This command is not currently supported.

Result Code	Description
OK <pintype>	Command Success.
ERROR <Error_code>	Command Fail.

Response Parameters:

PINType –

0 – variable pin

1 – fixed pin

AT command Ex: at+rsibt_getpintype?\r\n

Response: OK 1\r\n

2.3.2.17 User confirmation

Description: This will give the confirmation for the values sent by remote BT devices at the time of bonding.

Binary Payload Structure:

```
typedef struct rsi_bt_cmd_user_confirmation {
   UINT08  BDAddress[6];
   UINT08  Confirmation;
} RSI_BT_CMD_USER_CONFIRMATION;
```

AT command format:

at+rsibt_usrconfirmation=<BDAddress>,<Confirmation>\r\n

Parameters:

bd_addr: BD address of the remote BT device which send connection request.

confirmation:

0- NO. If both remote and local values are not same.

1- YES. If both remote and local values are same.

Response Payload:

There is no response payload for this command.

AT command Ex: at+rsibt_usrconfirmation= AA-BB-CC-DD-EE-FF,1\r\n

Response: OK\r\n

2.3.2.18 Pass key Request Reply

Description: The user passkey entry is used to respond on a user passkey entry request (UPER).

Binary Payload Structure:

```
typedef struct rsi_bt_cmd_passkey_reply {

   UINT08   BDAddress[6];

   UINT08   ReplyType;

   UINT08   Reserved;

   UINT32   Passkey;

} RSI_BT_CMD_PASSKEY_REPLY;
```

AT command format:

at+rsibt_usrpasskey=<BDAddress>,<ReplyType>,<Passkey>\r\n

Parameters:

BDAddress – Remote BD Address.

ReplyType –

0 – negitive reply

1 – positive reply

Passkey – Entered Passkey number in decimanl (range from 0 to 999999).

Response Payload:

There is no response payload for this command.

AT command Ex:

at+rsibt_usrpasskey= AA-BB-CC-DD-EE-FF,1,123456\r\n

Response:OK\r\n

2.3.2.19 Pincode Request Reply

Description: The user pincode entry is used to respond on a user pin code entry request (UPER). If we want to connect with remote device then we can respond with positive reply. Else send negative reply.

Binary Payload Structure:

```
typedef struct rsi_bt_cmd_pincode_reply {

   UINT08   BDAddress[6];
```

```
  UINT08   ReplyType;

  UINT08   Reserved;

  UINT08   Pincode[MAX_PINCODE_REPLY_SIZE];

} RSI_BT_CMD_PINCODE_REPLY;
```

AT command format:

at+rsibt_usrpincode=<BDAddress>,<ReplyType>,<Pincode>\r\n

Parameters:

BDAddress – Remote BD Address.

ReplyType –

0 – negative reply

1 – positive reply

Reserved - Padding

Pincode – Entered Pincode number(must be in string format max string length is 16 bytes).

Response Payload:

There is no response payload for this command.

AT command Ex: at+rsibt_usrpincode= AA-BB-CC-DD-EE-FF,1,1234\r\n

Response: OK\r\n

2.3.2.20 Get Local Device Role

Description: This gets the role of the local BT module when connected with a particular remote BT device, based on BD address of the remote BT device.

Binary Payload Structure:

```
typedef struct rsi_bt_cmd_query_role {

  UINT08  BDAddress[6];

} RSI_BT_CMD_QUERY_ROLE;
```

AT command format:

at+rsibt_getmasterslaverole=<BDAddress>?\r\n

Parameters:

BDAddress – Remote BD Address

Response Payload:

typedef struct rsi_bt_resp_query_role {

UINT08 Role;

} RSI_BT_RESP_QUERY_ROLE;

Result Code	Description
OK <role>	Command Success with valid response.
ERROR <Error_code>	Command Fail.

Response Parameters:

Role –

0 – Master role

1 – slave role

AT command Ex: at+rsibt_getmasterslaverole= AA-BB-CC-DD-EE-FF?\r\n

Response: OK 1\r\n

2.3.2.21 Set Local Device Role or switch the role[1]

Description: This is used to change the current role of the local BT module, with respect to the remote BT device.

Binary Payload Structure:

typedef struct rsi_bt_cmd_set_role {

UINT08 BDAddress[6];

UINT08 Role;

} RSI_BT_CMD_SET_ROLE;

AT command format:

at+rsibt_setmasterslaverole=<BDAddress>,<Role>\r\n

Parameters:

BDAddress – Remote BD Address

Role - 0 – Master role

[1] This command is not currently supported.

1 – Slave role

Response Payload:

There is no response payload for this command.

AT command Ex:

at+rsibt _setmasterslaverole=AA-BB-CC-DD-EE-FF,1\r\n

Response: OK\r\n

2.3.2.22 Get Service List

Description: This is used to search for the services supported by the remote BT device.

Binary Payload Structure:

```
typedef struct rsi_bt_cmd_query_services {
   UINT08  BDAddress[6];
        } RSI_BT_CMD_QUERY_SERVICES;
```

AT command format:

at+rsibt_getsrvs=<BDAddress>\r\n

Parameters:

BDAddress – Remote BD Address

Response Payload:

```
typedef struct rsi_bt_resp_query_services {
   UINT08   NumberOfServices;
   UINT08  Reserved[3];
   UINT32  ServiceUUIDs[32];
} RSI_BT_RESP_QUERY_SERVICES;
```

Result Code	Description
OK <bd_addr>,<nbr_srvs_found>,<srv_uuid_1>,<srv_uuid_2>,…….	Command Success with valid response.
ERROR <Error_code>	Command Fail.

Parameters:

NumberOfServices – Number of services in the list

ServiceUUIDs – list of serivce UUID's

NOTE: it will Display only 32 bit UUID's

AT command Ex: at+rsibt_getsrvs= AA-BB-CC-DD-EE-FF \r\n

Response: OK\r\n

2.3.2.23 Search Service

Description: This is used to find whether a particular service is supported by the remote BT device.

Binary Payload Structure:

```
typedef struct {
   UINT08   BDAddress[6];
   UINT08 Reserved[2];
   UINT32   ServiceUUID;
} RSI_BT_CMD_SEARCH_SERVICE;
```

AT command format:

at+rsibt_searchsrv=<BDAddress>,<ServiceUUID>\r\n

Parameters:

BDAddress – Remote BD Address

ServiceUUID – 16 bit or 32 bit UUID.

Response Payload:

```
typedef struct  {
        UINT08  SearchStatus;
} RSI_BT_RESP_SEARCH_SERVICE;
```

Result Code	Description
OK <search_result>	Command Success with valid response.
ERROR <Error_code>	Command Fail.

Response parameters:

Search status: 1-Yes, 0-No

AT command Ex: at+rsibt_searchsrv= AA-BB-CC-DD-EE-FF,1105\r\n

Response: OK 1\r\n

2.3.2.24 Linkkey Reply

Description: The link key reply is used to respond on a link key request event. If we have previous link key of connecting device then we can respond with positive reply. Else send negative reply.

Binary Payload Structure:

```
typedef struct {
   UINT08   BDAddress[6];
   UINT08   ReplyType;
     UINT08   Reserved;
   UINT08   LinkKey[16];
} RSI_BT_CMD_LINKKEY_REPLY;
```

AT command format:

at+rsibt_usrlinkkey=<BDAddress>,<ReplyType>,<LinkKey>\r\n

Parameters:

BDAddress – Remote BD Address.

ReplyType –

0 – negitive reply

1 – positive reply

LinkKey – Link key saved for the remote BD address in host.

Response Payload:

There is no response payload for this command

ATcommandEx:at+rsibt_usrlinkkey=AA-BB-CC-DD-EE-FF,1,3C,A5,50,25,DC,D0,B0,AB,B7,C3,4F,4D,9,79,2C,5C\r\n

Response: OK\r\n

2.3.2.25 Set SSP mode

Description: Set SSP mode is used to enable Simple Secure Pair mode and also used to select the IOCapability for SSP mode.

Binary Payload Structure:

```
typedef struct rsi_bt_cmd_set_ssp_mode {
                UINT08   PairMode;
                UINT08   IOCapability;
```

} RSI_BT_CMD_SET_SSP_MODE;

AT command format:

at+rsibt_setsspmode=<PairMode>,<IOCapability>\r\n

Parameters:

PairMode–

0 – Disable

1 - Enable

IOCapability –

0x00 - DisplayOnly

0x01 - DisplayYesNo

0x02 - KeyboardOnly

0x03 - NoInputNoOutput

Response Payload:

There is no response payload for this command

ATcommandEx:

at+rsibt_setsspmode=1,1\r\n

Response: OK\r\n

2.3.2.26 Sniff Mode

Description: Enables the Host to support a low-power policy and allows the devices to enter Inquiry Scan, Page Scan, and a number of other possible actions.

The local device will return the actual sniff interval in the Interval parameter of the Mode Change event, if the command is successful.

Binary Payload Structure:

```
typedef struct {
   UINT08  BDAddress[RSI_BT_BD_ADDR_LEN];
   UINT16  SniffMaxIntr;
   UINT16  SniffMinIntr;
   UINT16  SniffAttempt;
   UINT16  SniffTimeout;
 }RSI_BT_CMD_SNIFF_MODE;
```

AT command format:

at+rsibt_sniffmode=<BDAddress>,<SniffMaxIntr>,<SniffMinIntr>,<Sniff Attempt>,<sniffTimeout>\r\n

Parameters:

BDAddress- Remote BD Address.

SniffMaxIntr & SniffMinIntr- The Sniff_Max_Interval and Sniff_Min_Interval command parameters are used to specify the requested acceptable maximum and minimum periods in the Sniff Mode.

SniffAttempt- Master shall poll the slave at least once in the sniff attempt transmit slots starting at each sniff anchor point.

SniffTimeout- Timeout after which device enter sniff subrating mode.

Response Payload:

There is no response payload for this command

ATcommandEx:

at+rsibt_sniffmode= AA-BB-CC-DD-EE-FF ,192,160,4,2\r\n

Response: OK\r\n

2.3.2.27 Sniff Exit

Description: To end the Sniff mode.

Binary Payload Structure:

```
typedef struct {
   UINT08   BDAddress[RSI_BT_BD_ADDR_LEN];
 }RSI_BT_CMD_SNIFF_EXIT;
```

AT command format:

at+rsibt_sniffexit=<BDAddress>\r\n

Parameters:

BDAddress- Remote BD Address.

Response Payload:

There is no response payload for this command

ATcommandEx:

at+rsibt_sniffexit= AA-BB-CC-DD-EE-FF \r\n

Response: OK\r\n

2.3.2.28 Sniff Subrating[1]

Description: When the sniff mode timeout has expired a device shall enter sniff subrating mode. Sniff subrating mode allows a device to use a reduced number of sniff anchor points.

Binary Payload Structure:

```
typedef struct {
   UINT08  BDAddress[RSI_BT_BD_ADDR_LEN];
   UINT16  MaxLatency;
   UINT16  MinRemoteTimeout;
   UINT16  MinLocalTimeout;
 }RSI_BT_CMD_SNIFF_SUBRATING;
```

AT command format:

at+rsibt_sniffsubrating=<BDAddress>,<MaxLatency>,< MinRemoteTimeout >,< MinLocalTimeout >\r\n

Parameters:

BDAddress- Remote BD Address.

maximum_latency- Maximum allowed sniff subrate of the remote device.

minimum_remote_timeout- Minimum base sniff subrate timeout that the remote device may use

minimum_local_timeout- Minimum base sniff subrate timeout that the local device may use.

Response Payload:

There is no response payload for this command

ATcommandEx:

at+rsibt_sniffsubrating= AA-BB-CC-DD-EE-FF , 192,1000,1000\r\n

Response: OK\r\n

2.3.3 SPP commands

Note: spp profile will not connect without pair process or authentication.

2.3.3.1 SPP Connect

Description: This is used to establish SPP connection with the remote BT device, specified by the BD address.

[1] Currently, the sniff subrating command is not supported.

Binary Payload Structure:

```
typedef struct rsi_bt_cmd_spp_connect {
   UINT08  BDAddress[6];
} RSI_BT_CMD_SPP_CONNECT;
```

AT command format:

at+rsibt_sppconn=<BDAddress>\r\n

Parameters:

BDAddress – Remote BD address.

Response Payload:

There is no response payload for this command.

AT command Ex: at+rsibt_sppconn= AA-BB-CC-DD-EE-FF\r\n

Response: OK\r\n

2.3.3.2 SPP Disconnect

Description:

This is used to disconnect the SPP connection with the remote BT device.

Binary Payload Structure:

```
typedef struct rsi_bt_cmd_spp_disconnect {
   UINT08  BDAddress[6];
} RSI_BT_CMD_SPP_DISCONNECT;
```

AT command format:

at+rsibt_sppdisconn=<BDAddress>\r\n

Parameters:

BDAddress – Remote BD address

Response Payload:

There is no response payload for this command.

AT command Ex: at+rsibt_sppdisconn= AA-BB-CC-DD-EE-FF \r\n

Response: OK\r\n

2.3.3.3 SPP Transfer

Description: This is used to send data to the remote BT device using SPP profile. This command contains a data length field, which tells the BT module about the length of data in bytes user want to send from the application.

Binary Payload Structure:

```
typedef struct rsi_bt_cmd_spp_transfer {
   UINT16  DataLength;
   UINT08  Data[200];
} RSI_BT_CMD_SPP_TRANSFER;
```

AT command format:

at+rsibt_spptx=<DataLength><Data>\r\n

Parameters:

DataLength – SPP data length (range of Data length is 1 to 200 Bytes).

Data – SPP data.

Response Payload:

There is no response payload for this command.

AT command Ex: at+rsibt_spptx=5,iiiii\r\n

Response: OK\r\n

2.3.4 IAP commands

This profile is used for remote controlling of apple devices. It is having IAP1 and IAp2 protocols. Currently, we are supporting only IAP1. We are supporting two lingos are General lingo and Simple remote lingo.

2.3.4.1 IAP connect

Description: This command is used to establish an IAP connection with the remote Apple device, specified by the BD address.

Command:

at+rsibt_iapconn=<bd_addr>\r\n

Parameters:

bd_addr- BD address of the remote Apple device, with which the IAP connection has to be established.

Response:

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:

at+rsibt_iapconn=11-22-33-44-55-66\r\n

Response:

OK\r\n

2.3.4.2 IAP Disconnect

Description: This command is used to disconnect the Apple device which was connected using IAP.

Command:

at+rsibt_iapdisconn=<bd_addr>\r\n

Parameters:

bd_addr- BD address of the remote Apple device, with which the IAP connection has to be released.

Response:

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:

at+rsibt_iapdisconn=11-22-33-44-55-66\r\n

Response:

OK\r\n

2.3.4.3 IAP Set Accessory Information

Description: This command is used to set the accessory information like name, manufacturer, model number, serial number, firmware and hardware version, supported languages by the accessory and currently used language in the accessory. This information should be sent to the accessory before starting the identification procedure without fail.When the user wants to change these values, the accessory must go through the identification procedure again, which will update the accessory information at Apple device side. This command is common for both IAP1 and IAP2.

Command:

at+rsibt_iapsetaccessoryinfo=<info_type>,<data_len>,<info_data>\r\n

Parameters:

nbr_lang_supp- This parameter is used only for Info type 8, which indicates the number of languages supported by the accessory.

Info_Type	Info_Data
1	**Accessory Name.** This should be <= 63 characters.
2	**Accessory Manufacturer.** This should be <= 63 characters.
3	**Accessory Model Number.** This should be <= 63 characters.
4	**Accessory Serial Number.** This should be <= 63 characters.
5	**Accessory Firmware Version.** Ex: Major version, Minor version, Revision version.
6	**Accessory Hardware Version.** Ex: Major version, Minor version, Revision version.
7	**Accessory Current Language.**
8	**Accessory Supported Languages.** These values should be according the following format mentioned in the below link. **http://www.loc.gov/standards/iso639-2/php/English_list.php**

Response:

Result Code	Description

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:

at+rsibt_iapsetaccessoryinfo=1,Redpine_BT_Accessory\r\n

at+rsibt_iapsetaccessoryinfo=5,1,2,3\r\n

at+rsibt_iapsetaccessoryinfo=7,1,en\r\n

at+rsibt_iapsetaccessoryinfo=8,3,en,fr,hi\r\n

Response:

OK\r\n

2.3.4.4 IAP Find Protocol Type

Description: This command is used to find the type of protocol(IAP1 or IAP2)supported by the connected Apple device. If the connected device supports IAP1, the identification procedure will start automatically. If it doesn't start, start IAP1 identification manually, by sending the IAP1 Identification command. If the device supports IAP2, the IAP2 Identification procedure should be started manually by sending IAP2 Identification command.

Command:

at+rsibt_iapfindprotocoltype\r\n

Parameters:

None

Response:

Result Code	Description
OK <type>	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:

at+rsibt_iapfindprotocoltype\r\n

Response:

OK 2\r\n

2.3.4.5 IAP Set Protocol Type

Description: This command is used to Set the protocol type of accessory to IAP1. This command is used when the user wants to use only IAP1 to communicate with the device, even though the device supports IAP2. This command forces the accessory to use only IAP1. IAP1 Identification should be done manually. This command is used to set only IAP1 in the accessory, but not used to set IAP2.

Note: when this command is used, IAP Find Protocol Type command should not be used prior to this command. If IAP Find Protocol Type command is already used, the Bluetooth connection has to be re-established, to use this (IAP Set Protocol Type) command.

Command:

at+rsibt_iapsetprotocoltype=1\r\n

Parameters:

Type: 1 – IAP1
2 – IAP2

Response:

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:

at+rsibt_iapsetprotocoltype=1\r\n

Response:

OK\r\n

2.3.4.6 IAP Set Application Protocol Information

Description: The Application Protocol Information is needed only when the accessory wants to communicate with an application in the Apple device.

Using this command, the accessory will intimate the Apple device about the details of application. This must be done before identification procedure. Using this command the user can disable the support of communicating with the application in Apple device. This command is common for both IAP1 and IAP2.

Command:

at+rsibt_iapsetappprotocolinfo=<mode>,<protocol_index>,<protocol_str_len>,<protocol_str>,<bundle_seed>,<meta_data>\r\n

Parameters:

Mode - 0 – Disable application support in Accessory.
1 – Enable application support in Accessory.

If this value is **"0",** the accessory will neglect the remaining fields in the command and removes the application support in accessory.

protocol_index - Index of the protocol assigned by the accessory. 1 to 255 can be used.

protocol_str_len **-** Length of the protocol string.

protocol_str- Actual protocol string. This is a reverse DNS name like **"com.apple.Music"**

bundle_seed- Bundle seed ID string allocated by Apple.Inc, to the accessory application developer.

meta_data	Details
0	The Apple device doesn't try to find a matching app. It doesn't display a "Find App For This Accessory" button in the Settings > General .About >Accessory Name screen.
1	The Apple device tries to find a matching app. It doesn't display a "Find App For This Accessory" button in the Settings > General .About >Accessory Name screen.
3	The Apple device doesn't try to find a matching app, but it displays a "Find App For This Accessory" button in the Settings > General .About >Accessory Name screen so the user can find one.

Response:

Result Code	Description

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:

at+rsibt_iapsetappprotocolinfo\r\n

Response:

OK\r\n

2.3.4.7 IAP1 Identification

Description: This command is used to start the Identification procedure of an accessory with the Apple device. The Identification will be followed by the Authentication procedure, which will be initiated by the Apple device. Identification is the mandatory step to be followed by the accessory. User can use remaining commands only after receiving the notification

"AT+RSIBT_IAP1_ACCESSORY_AUTH_COMPLETED"

Command:

at+rsibt_iap1identification\r\n

Parameters:

NONE

Response:

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:

at+rsibt_iap1identification\r\n

Response:
OK\r\n

2.3.4.8 IAP1 Apple Device Authentication

Description: This command is used to authenticate the connected Apple device. This procedure will be initiated by the accessory and it is not mandatory.

Command:

at+rsibt_iap1deviceauthentication\r\n

Parameters:

NONE

Response:

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:
at+rsibt_iap1deviceauthentication\r\n

Response:
OK\r\n

2.3.4.9 Set Assistive Touch

Description: This command is used to enable or disable the assistive touch feature in the Apple device. The restore on exit parameter indicates, whether the Apple device has to restore the previous (before connected with the accessory) settings of the assistive touch feature, after the accessory gets disconnected with the Apple device.

Command:

at+rsibt_iap1setassistivetouch=<mode>,<restore_on_exit>\r\n

Parameters:

Mode- Assistive Touch Mode.
0 – OFF.
1 – ON.

restore_on_exit- 1 – Apple device restores the original settings when the accessory is disconnected.
0 – Apple device doesn't perform the restore.

Response:

Result Code	Description

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:
at+rsibt_iap1setassistivetouch=1,1\r\n

Response:
OK\r\n

2.3.4.10 Set Voice Over

Description: This command is used to enable or disable the voiceover feature in the Apple device. The restore on exit parameter indicates, whether the Apple device has to restore the previous (before connected with the accessory) settings of the voiceover feature, after the accessory gets disconnected with the Apple device.

Command:

at+rsibt_iap1setvoiceover=<mode>,<restore_on_exit>\r\n

Parameters:

Mode- Voice over Mode.
0 – OFF.
1 – ON.
restore_on_exit - 1 – Apple device restores the original settings when the accessory is disconnected.
0 – Apple device doesn't perform the restore.

Response:

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:
at+rsibt_iap1setvoiceover=1,1\r\n

Response:

OK\r\n

2.3.4.11 IAP1 Get Ipod Info (Name, SW version, serial number)

Description: This command is used to get the information (name, software version, serial number) of the connected Apple device.

Command:

at+rsibt_iap1getipodinfo\r\n

Parameters:

Name- Name of the Apple device connected. It is null terminated character array.

Software Version- Software version of the connected Apple device. Major, Minor and Revision version numbers separated by "-".

Serial Number- Serial Number of the connected Apple Device. It is null terminated character array.

Response:

Result Code	Description
OK <Name(null terminated array)>,<software version>,<serial number>	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:

at+rsibt_iap1getipodinfo\r\n

Response:

OK Apple_iPhone,6-1-3,6Q105EN0A4S\r\n

2.3.4.12 IAP1 Set Extended Interface Mode (ON/OFF)

Description: This command is used to set the Extended Interface Mode ON and OFF in the connected Apple device. This command works only when the Accessory supports Extended Interface Mode Lingo.

Command:

at+rsibt_iap1setextendedintfmode=<mode>\r\n

Parameters:

Mode- Sets the mode of Extended Interface.
0 – Extended Interface Mode OFF
1 – Extended Interface Mode ON.

Response:

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:
at+rsibt_iap1setextendedintfmode=1\r\n

Response:
OK\r\n

2.3.4.13 IAP1 Get Lingo Protocol Version

Description: This command is used to get the protocol version of a particular lingo, supported by the Apple device.

Command:

at+rsibt_iap1getlingoprotocolversion=<lingo_id>\r\n

Parameters:

lingo_id- Lingo ID for which protocol version has to be known.

Protocol version- Protocol version of the lingo. Major and minor version will be separated by "-".

Response:

Result Code	Description

Result Code	Description
OK <lingo_id>,<protocol version>	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:
at+rsibt_iap1getlingoprotocolversion=0\r\n

Response:
OK 0,1-3\r\n

2.3.4.14 IAP1 Set Ipod Preferences

Description: This command is used to set the IPOD class of preference, like video out settings, screen configuration etc.

Command:

at+rsibt_iap1setipodpreferences=<class_id>,<setting_id>,<restore_on_exit>\r\n

Parameters:

class_id- Class ID of the preference.
setting_id- Settings ID of the preference.
restore_on_exit- 1 – Restore original settings on exit (disconnection).
0 – Keep current settings even after exit.

class_id	Preference	Settings	Settings_id
0	Video out	OFF	**0**
		ON	**1**
1	Screen Configuration	Fill entire screen	**0**
		Fit to screen edge	**1**
2	Video Signal Format	NTSC	**0**
		PAL	**1**
3	Line-out usage	Not used	**0**
		Used	**1**
8	Video out connection	Composite	**1**
		S-Video	**2**
		Component	**3**
9	Closed	OFF	**0**

	captioning	ON	**1**
10	Video monitor aspect ratio	4:3 (full screen)	**0**
		16:9 (wide screen)	**1**
12	Subtitles	OFF	**0**
		ON	**1**
13	Video alternate audio channel	OFF	**0**
		ON	**1**
15	Pause on power removal	OFF	**0**
		ON	**1**

Response:

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:
at+rsibt_iap1setipodpreferences=0,1,1\r\n

Response:
OK\r\n

2.3.4.15 IAP1 Get Ipod Preferences

Description: This command is used for Identification of an accessory with the Apple device.

Command:

at+rsibt_iap1getipodpreferences=<class_id>\r\n

Parameters:

Class_id- Class ID of the preference.
Setting_id- Settings ID of the preference.

Response:

Result Code	Description
OK <class_id>,<setting_id>	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:
at+rsibt_iap1getipodpreferences=0\r\n

Response:
OK 0,1\r\n

2.3.4.16 IAP1 Set UI Mode

Description: This command is used to set the UI mode of the Apple device.

Command:

at+rsibt_iap1setuimode=<mode>\r\n

Parameters:

Mode - UI mode options.
- 0 – Standard Apple device operating mode
- 1 – Extended interface mode
- 2 – ipod out full screen mode
- 3 – ipod out action safe mode.

Response:

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:
at+rsibt_iap1setuimode=0\r\n

Response:
OK\r\n

2.3.4.17 IAP1 Get UI Mode

Description: This command is used to get the current UI mode set in the Apple device.

Command:

at+rsibt_iap1getuimode\r\n

Parameters:

Mode - UI mode options.
0 – Standard Apple device operating mode
1 – Extended interface mode
2 – ipod out full screen mode
3 – ipod out action safe mode.

Response:

Result Code	Description
OK <mode>	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:
at+rsibt_iap1getuimode\r\n

Response:
OK 0\r\n

2.3.4.18 IAP1 Set Event Notification

Description: This command is used to set selected notifications from the Apple device. These notifications are asynchronous events sent by the Apple device on occurrence of a particular event in Apple device.

Command:

at+rsibt_iap1seteventnotification=<nbr_notifications>,<1>,<2>,….<n>\r\n

Parameters:

nbr_notifications- Number of notifications to be set.

Notification	Details

2	Flow Control.
3	Radio tagging status.
4	Camera status.
5	Charging Info.
9	Database Changed.
10	Now Playing Application Bundle Name Status.
11	Session Space Available.
15	Ipod out mode status.
17	Bluetooth connection status
19	Now playing application display name
20	Assistive touch status.

Response:

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:

at+rsibt_iap1seteventnotification=6,2,3,4,5,9,10\r\n

Response:

OK\r\n

2.3.4.19 IAP1 Get Event Notification

Description: This command is used to get the list of notification enabled in the Apple device.

Command:

at+rsibt_iap1geteventnotification\r\n

Parameters:

nbr_notifications - Number of notifications already set in the Apple device.

Notification	Details
2	Flow Control.
3	Radio tagging status.
4	Camera status.
5	Charging Info.

9	Database Changed.
10	Now Playing Application Bundle Name Status.
11	Session Space Available.
15	Ipod out mode status.
17	Bluetooth connection status
19	Now playing application display name
20	Assistive touch status.

Response:

Result Code	Description
OK <nbr_notifications>,<notification1>,<2>,<3>,….	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:

at+rsibt_iap1geteventnotification\r\n

Response:

OK 6,2,3,4,5,9,10\r\n

2.3.4.20 IAP1 Get Supported Event Notification

Description: This command is used to get the list of notifications supported by Apple device.

Command:

at+rsibt_iap1suppeventnotifications\r\n

Parameters:

nbr_notifications- Number of notifications supported by Apple device.

Notification	Details
2	Flow Control.
3	Radio tagging status.
4	Camera status.
5	Charging Info.
9	Database Changed.
10	Now Playing Application Bundle Name Status.
11	Session Space Available.

15	Ipod out mode status.
17	Bluetooth connection status
19	Now playing application display name
20	Assistive touch status.

Response:

Result Code	Description
OK <nbr_notifications>,<notification1>,<2>,<3>,….	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:
at+rsibt_iap1suppeventnotifications\r\n

Response:
OK 6,2,3,4,5,9,10\r\n

2.3.4.21 IAP1 Launch Application

Description: This command is used to launch an application in the Apple device.

Command:

at+rsibt_iap1launchapplication=<app_name>\r\n

Parameters:

app_name- Name of the application to be launched. It is a null terminated character array.

Response:

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:
at+rsibt_iap1launchapplication=com.apple.Music\r\n

Response:

OK\r\n

2.3.4.22 IAP1 Get Localization Info

Description: This command is used to get the localization information (language and region) currently set in the Apple device.

Command:

at+rsibt_iap1getlocalizationinfo\r\n

Parameters:

Language- Current language used in the Apple device. It is null terminated character array.

Region- Region settings in the Apple device. It is null terminated character array.

Response:

Result Code	Description
OK <language>,<region>	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:

at+rsibt_iap1getlocalizationinfo\r\n

Response:

OK en-GB,en-IN\r\n

2.3.4.23 IAP1 Application Data Session Acknowledgment

Description: This command is used to acknowledge the requests (open and close data session) related to the data session and to acknowledge the received data packet from the Apple device.

Command:

at+rsibt_iap1appdatasessionack=<recv_cmd_type>,<status>\r\n

Parameters:

recv_cmd_type- Acknowledgment for the event received.

1 – Open Data Session.

2 – Close Data Session.

3 – IPod Data Transfer.

Status- 0 – Success.

4 – Bad Parameter. This error should be set only when the Apple device sends Ipod Data without opening a session.

Response:

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:

at+rsibt_iap1appdatasessionack=1,0\r\n

Response:

OK\r\n

2.3.4.24 IAP1 Application Accessory Data Transfer

Description: This command is used to transfer data to the Apple device. If an Apple device unable to handle data, it will return an error **"ERR_IAP1_DROPPED_DATA".** At that time the accessory has to wait for certain amount of time until it receives the event notification **"AT+RSIBT_IAP1_SESSION_SPACE_AVAILABLE".**

Command:

at+rsibt_iap1appaccessorydatatransfer=<session_id>,<data_len>,<data>\r\n

Parameters:

session_id- Session ID sent by the Apple device at the time of opening the data session.

data_len- Length of data sent by the Accessory to the Apple device.

data- Actual data sent by the Accessory.

bytes_dropped- Number of bytes dropped by the Apple device without handling.

Response:

Result Code	Description
OK	Command Success.
ERROR <Error_code>,<bytes_dropped>	Command Fail.

AT command Ex:
at+rsibt_iap1appaccessorydatatransfer=1,7,welcome\r\n

Response:
OK\r\n
ERROR 8017,50\r\n

2.3.4.25 IAP1 Get Voice Over Parameter

Description: This command is used to get the current value of a particular voice over parameters, like volume, speaking rate.

Command:

at+rsibt_iap1getvoiceoverparameter=<param_type>\r\n

Parameters:

param_type- Type of the parameter to be set.
param_value- Parameter value corresponding to the parameter type.

Param_Type	Param_Value
0	Voiceover volume. 0(mute) to 255(full volume).
1	Speaking Rate. 0(slow) to 255(fast).

Response:

Result Code	Description
OK <param_type>,<param_value>	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:
at+rsibt_iap1getvoiceoverparameter=0\r\n

Response:

OK 0,150\r\n

2.3.4.26 IAP1 Set VoiceOver Parameter

Description: This command is used to set the value of a particular voice over parameters, like volume, speaking rate.

Command:

at+rsibt_iap1setvoiceoverparameter<param_type>,<param_value>\r\n

Parameters:

param_type- Type of the parameter to be set.

param_value- Parameter value corresponding to the parameter type.

Param_Type	Param_Value
0	Voiceover volume. 0(mute) to 255(full volume).
1	Speaking Rate. 0(slow) to 255(fast).

Response:

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:
at+rsibt_iap1setvoiceoverparameter=0,150\r\n

Response:

OK\r\n

2.3.4.27 IAP1 Set VoiceOver Context

Description: This command is used to set the context of voiceover in which the user is currently working.

Command:

at+rsibt_iap1setvoiceovercontext=<param_type>\r\n

Parameters:

param_type- The user interface context to be set.

Param_type	Details
0	None. Exit from or disable any context.
1	Header.
2	Link.
3	Form.
4	Cursor.
5	Vertical Navigation. Sets Move Next/Move Previous to move down/up to the next item vertically
6	Value Adjustment. Sets Move Next/Move Previous to increment/decrement the value of the current item by 5%.
7	Zoom Adjustment. Sets Move Next/Move Previous to zoom the current item in/out by 1 increment.

Response:

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:
at+rsibt_iap1setvoiceovercontext=1\r\n

Response: OK\r\n

2.3.4.28 IAP1 VoiceOver Event

Description: This command is used to send the voice over events, like Move to first, move to last, scroll page up/down etc to control the Apple device.

Command:

at+rsibt_iap1voiceoverevent=<event_type>\r\n

Parameters:

event_type	Details
1	Move to First.
2	Move to Last.
3	Move to Next. When option 5 is set in the **"Set Voiceover Context"** command, Move to Next and Move to Previous are used to navigate up and down respectively.
4	Move to Previous.
5	Scroll Left Page.
6	Scroll Right Page.
7	Scroll Up Page.
8	Scroll Down Page.
11	Cut.
12	Copy.
13	Paste.
14	Home
18	Pause Speaking.
19	Resume Speaking.
20	Read all text from current point.
21	Read all text from top.
23	Escape. Voice over exit from a modal view, such as an alert or popover.

Response:

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:
at+rsibt_iap1voiceoverevent=1\r\n

Response:
OK\r\n

2.3.4.29 IAP1 VoiceOver Text Event

Description: This command is used to send text to Apple device, which can be used in filling a form, entering URL, and also used as text-to-speech, which can be pronounced by the Apple device.

Command:

at+rsibt_iap1voiceovertextevent=<event_type>,<data_len>,<data>\r\n

Parameters:

event_type- Type of the event to be set.
10 – Text in this type is used to fill the forms, links etc.
22 – Text in this type is used to be pronounced by the Apple device like text-to-speech.

data_len- Length of the event data.

data- Event data corresponding to the event type.

Response:

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:
at+rsibt_iap1voiceovertextevent=22,29,Welcome to RedpineSignals.\r\n

at+rsibt_iap1voiceovertextevent=10,14,www.google.com\r\n

Response:
OK\r\n

2.3.4.30 IAP1 VoiceOver Touch Event

Description: This command is used to tap a selected item (select an item) in the Apple device.

Command:

at+rsibt_iap1voiceovertouchevent=<event_type>\r\n

Parameters:

event_type- Type of the event to be set. Set **0** as event type.

Response:

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:
at+rsibt_iap1voiceovertouchevent=0\r\n

Response:
OK\r\n

2.3.4.31 IAP1 Current VoiceOver Value

Description: This command is used to get the value of current voiceover item like the percentage of volume set.

Command:

at+rsibt_iap1currvoiceovervalue\r\n

Parameters:

Value- Value like volume %.

Response:

Result Code	Description
OK <value>	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:
at+rsibt_iap1currvoiceovervalue\r\n

Response:
OK 62%\r\n

2.3.4.32 IAP1 Current VoiceOver Hint

Description: This command is used to get the hint of current voiceover item like the percentage of volume set.

Command:

at+rsibt_iap1currvoiceoverhint\r\n

Parameters:

None

Response:

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:
at+rsibt_iap1currvoiceoverhint\r\n

Response:
OK\r\n

2.3.4.33 IAP1 Current VoiceOver Trait

Description: This command is used to get the type of item currently pointed by voice over.

Command:

at+rsibt_iap1currvoiceovertrait\r\n

Parameters:

Trait_type	Details
0	Button
1	Link
2	Search

3	Image
4	Selected
5	Sound
6	Keyboard Key
7	Static Text
8	Summary Element
9	Not Enabled
10	Updates Frequently
11	Starts Media Session
12	Adjustable
13	Back Button
14	Maps
15	Delete Key

Response:

Result Code	Description
OK <trait_type>	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:
at+rsibt_iap1currvoiceovertrait\r\n

Response:
OK 0,13\r\n

2.3.4.34 IAP1 iPod Out Button

Description: This command is used to set the voice over events, like Move to first, move to last, scroll page up/down etc.

Command:

at+rsibt_iap1ipodoutbutton=<button_src>,<button_type>\r\n

Parameters:

button_src	Description
0	Car center console
1	Steering wheel

2	Car dashboard

button_type	Description
0	Select event
1	Left event
2	Right event
3	Up event
4	Down event
5	Menu button event

Response:

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:
at+rsibt_iap1ipodoutbutton=1,1\r\n

Response:
OK\r\n

2.3.4.35 IAP1 Video Button

Description: This command is used to access Video controls in the Apple device.

Command:

at+rsibt_iap1videobutton=<button_type>\r\n

Parameters:

button_type- Type of button pressed among the buttons mentioned below

Button_Type	Details
0	Play/Pause
1	Next Video
2	Previous Video
3	Stop

4	Play/Resume
5	Pause
6	Begin FF
7	Begin REW
8	Next Chapter
9	Previous Chapter
10	Next Frame
11	Previous Frame

Response:

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:
at+rsibt_iap1videobutton=1\r\n

Response:

OK\r\n

2.3.4.36 IAP1 Audio Button

Description: This command is used to access Audio controls in the Apple device.

Command:

at+rsibt_iap1audiobutton=<button_type>\r\n

Parameters:

button_type- Type of button pressed among the buttons mentioned below.

Button_Type	Details
0	Play/Pause
3	Next Track
4	Previous Track
5	Next Album

6	Previous Album
7	Stop
8	Play/Resume
9	Pause
11	Next Chapter
12	Previous Chapter
13	Next Playlist
14	Previous Playlist
15	Shuffle Setting Advance
16	Repeat Setting Advance
17	Begin Fast Forward
18	Begin Rewind
19	Menu

Response:

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:
at+rsibt_iap1audiobutton=4\r\n

Response:

OK\r\n

2.3.4.37 IAP1 Context Button

Description: This command is used to access the media controls according to the current type of media running in the Apple device. For example, when audio is playing in the Apple device, this command can be able to control the audio player. Like this it can control the video player, app store player as well.

Command:

at+rsibt_iap1contextbutton=<button_type>\r\n

Parameters:

button_type- Type of button pressed among the buttons mentioned below.

Button_Type	Details
0	Play/Pause

3	Next Track
4	Previous Track
5	Next Album
6	Previous Album
7	Stop
8	Play/Resume
9	Pause
11	Next Chapter
12	Previous Chapter
13	Next Playlist
14	Previous Playlist
15	Shuffle Setting Advance
16	Repeat Setting Advance
17	Power ON
18	Power OFF
19	Backlight for 30 Seconds
20	Begin Fast Forward
21	Begin Rewind
22	Menu
23	Select
24	Up Arrow
25	Down Arrow
26	Backlight OFF

Response:

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:
at+rsibt_iap1contextbutton=0\r\n

Response:
OK\r\n

2.3.4.38 IAP1 Radio Button

Description: This command is used to initiate tagging action by iPod's Radio Application. This is supported only in 5G nano iPod.

Command:

at+rsibt_iap1radiobutton=<button_status>\r\n

Parameters:

button_status	Description
0	Radio button pushed to tag current song.
2	Button released.

Response:

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:

at+rsibt_iap1radiobutton=2\r\n

Response:
OK\r\n

2.3.4.39 IAP1 Camera Button

Description: This command is used to control the camera in the Apple device. This command can be used only when the accessory get connected with ipod 5G.

Command:

at+rsibt_iap1camerabutton=<button_status>\r\n

Parameters:

button_status	Description
0	Button up
1	Button down

Response:

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:

at+rsibt_iap1camerabutton=1\r\n

Response:

OK\r\n

2.3.4.40 IAP1 Rotation Input

Description: This command is used to set the voice over events, like Move to first, move to last, scroll page up/down etc.

Command:

at+rsibt_iap1rotationinput=<usr_action_dur>,<src>,<dir>,<action>,<type>,<nbr_moves>,<total_moves>\r\n

Parameters:

usr_action_dur - Number of milliseconds since the start of the current user action.

Src - Location of the wheel.

0 – Car center console.

1 – Steering wheel.

2 – Car dashboard.

Dir - 0 – Counter clockwise.

1 – Clockwise.

Action - 0 – Rotation action completed; user has released control.

1 – Rotation in progress; wheel turning.

2 – Rotation repeat; the user has not advanced the wheel from the position reported in the last RotationInputStatus packet.

Type - 0 – Wheel has detents.

1 – Wheel reports angular degrees.

Nbr_moves - Number of detents or degrees the user has moved the wheel since the last RotationInputStatus packet.

Total_moves - Constant number of detents or degrees in a full turn of the wheel (max 360).

Response:

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:

at+rsibt_iap1rotationinput=100,0,0,1,0,5,20\r\n

Response:
OK\r\n

2.3.4.41 IAP1 Register HID Report Descriptor

Description: This command is used to register a HID Report descriptor with the Apple device. The format of the descriptor is same that is defined in the USB specification.

Command:

at+rsibt_iap1reghiddescriptor=<desc_index>,<vendor_id>,<product_id>,<country_code>,<len>,<descriptor>\r\n

Parameters:

desc_index: HID descriptor index.
vendor_id: Vendor ID of the accessory.
product_id: Product ID of an accessory.
country_code: country code. Country codes are listed in the table below.
len: Length of the descriptor bytes.
descriptor: Report descriptor. The descriptor values are not the ASCII values, but the numerical values.

Response:

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:

Here is the example for mouse report descriptor.

at+rsibt_iap1reghiddescriptor=0,1452,4626,20,50, 05 01 09 02 A1 01 09 01 A1 00 05 09 19 01 29 03 15 00 25 01 95 03 75 01 81 02 95 01 75 05 81 01 05 01 09 30 09 31 15 81 25 7F 75 08 95 02 81 06 C0 C0 \r\n

Response:

OK\r\n

Apple also provides some customer usage page (0x0C), to access some of the controls in the Apple devices. The control codes are listed below.

HID Usage ID	HID Usage Name	iOS Function
0x0030	Power	Lock
0x0040	Menu	Home
0x00B5	Scan Next Track	Transport Right
0x00B6	Scan Previous Track	Transport Left
0x00CD	Play/Pause	Play/Pause
0x00E2	Mute	Mute
0x00E9	Volume increment	Louder
0x00EA	Volume decrement	Softer
0x01AE	AL keyboard layout	Toggle onscreen keyboard
0x01B1	AL screen saver	Picture frame
0x0221	AC search	Spotlight

2.3.4.42 IAP1 Send HID Report

Description: This command is used to send a report to the Apple device based on the descriptor registered with the Apple device.

Command:

at+rsibt_iap1sendhidreport=<desc_index>,<report_type>,<len>,<report>\r\n

Parameters:

desc_index- HID descriptor index, already registered with the Apple device.

report_type- Input report.

Len- Length of the report in bytes.

Report- Device report. The report values are not the ASCII values, but the numerical values.

Response:

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:

Here is the example for mouse report.
at+rsibt_iap1sendhidreport=0,0,3, 01 00 00 \r\n

Response:
OK\r\n

2.3.4.43 IAP1 Unregister HID Report Descriptor

Description: This command is used to unregister a HID Report descriptor with the Apple device.

Command:

at+rsibt_iap1unreghiddescriptor=<desc_index>\r\n

Parameters:

desc_index: HID descriptor index, already registered with the Apple device.

Response:

Result Code	Description
OK	Command Success.
ERROR <Error_code>	Command Fail.

AT command Ex:

at+rsibt_iap1unreghiddescriptor=0\r\n

Response:

OK\r\n

2.3.5 PER Commands

2.3.5.1 PER Transmit

Description: This will allow the configuration of the following parameters and starts the transmission of packets.

Binary Payload Structure:

typedef struct rsi_bt_cmd_per_transmit{

UINT08 type;

UINT08 enable;

UINT08 BDAddress[6];

UINT16 pktlength;

UINT08 pkttype;

UINT08 linktype;

UINT08 rate;

UINT08 rx_channel;

UINT08 tx_channel;

UINT08 scramblerseed;

UINT32 nbrofpkts;

UINT08 payloadtype;

UINT08 protocolmode;

UINT08 lechanneltype;

UINT08 powerindex;

UINT08 transmitmode;

UINT08 freqhopenable;

UINT08 antennaselect;

}RSI_BT_CMD_PER_TRANSMIT;

Parameters:

<type>: This parameter indicates that this structure belongs to transmit

mode. This type has value 4.

<enable>: This parameter indicates that this structure belongs to transmit mode. „0“ -disable, „1“ -enable

<BDAddress>: This is the device address in Classic mode and Access Code in LE mode. It is a 48-bit address in hexadecimal format with colon separation. e.g., 00:23:A7:01:02:03.

<pktlength>: This is the length of the packet, in bytes, to be transmitted. Range is from 1 to 1021.

<pkttype>: This is the type of packet to be transmitted, as per the Bluetooth standard. Range is from 1 to 15.

<linktype>: This parameter indicates the Link Type – ACL, SCO, eSCO. This parameter is valid only in the Classic mode and invalid in LE mode. „0“ – SCO
, „1“ – ACL , „2“ – eSCO.

<rate>: This parameter decides whether the transmission has to happen in Basic Data Rate or Enhanced Data Rate in Classic mode. This parameter is invalid in LE mode. „1“ – Basic Data Rate , „2“ or „3“ – Enhanced Data Rate

<rx_channel>: This parameter indicates the Receive channel index, as per the Bluetooth standard.

<tx_channel>: This parameter indicates the Transmit channel index, as per the Bluetooth standard.

<scramblerseed>: This parameter is the initial seed to be used for whitening. It should be set to „0“ to disable whitening.

<nbrofpkts>: This is the number of packets to be transmitted. This is valid only when the <tx_mode> is set to Burst mode (0).

<payloadtype>: This parameter indicates the type of payload to be transmitted.

„0“ – Payload consists of zeros.

„1“ – Payload consists of 0xFF“ s.

„2“ – Payload consists of 0x55“ s

„3“ – Payload consists of 0xF0“ s.

„4“ – Payload consists of PN9 sequence.

<protocolmode>: This parameter is used to choose between Bluetooth Classic and LE modes for the packet transmission. „1“ – Classic mode , „2“ – LE Mode

<lechanneltype>: This parameter indicates the channel type in LE mode. „0“ – Advertising channel , „1“ – Data channel

<powerindex>: This is the transmit power (in dBm) to be used by the module. The value should be between 0 and 18.

<transmitmode>: This parameter is used to choose between Burst and Continuous modes of transmission. „0“ – Burst mode , „1“ – Continuous mode

<freqhopenable>: This parameter is used to choose the hopping pattern „0“ – No hopping , „1“ – Fixed hopping , „2“ – Random hopping

<antennaselect>: This parameter is used to select one of the two RF ports connecting to antennas. For the modules without integrated antenna, it is used to select between pins RF_OUT_1 and RF_OUT_2. For the modules with integrated antenna and U.FL connector, it used to select between the two.
„2“ – RF_OUT_2/Antenna , „3“ – RF_OUT_1/U.FL

2.3.5.2 Per receive

Description:

This will allow the configuration of the following parameters and starts the reception of packets.

Binary Payload Structure:

```
typedef struct
        rsi_bt_cmd_per_receive{
        UINT08  type;
        UINT08  enable;
        UINT08  BDAddress[6];
         UINT16 pktlength;
         UINT08 pkttype;
        UINT08  linktype;
        UINT08 rate;
```

```
        UINT08 rx_channel;

        UINT08  tx_channel;

        UINT08 scramblerseed;

        UINT08 payloadtype;

        UINT08 protocolmode;

        UINT08 lechanneltype;

        UINT08 freqhopenable;

        UINT08  antennaselect;

}RSI_BT_CMD_PER_RECEIVE;
```

Parameters:

<type>: This parameter indicates that this structure belongs to receive mode. This type has value 5.

<enable>: This parameter is used to enable or disable the receive mode. „0“ -disable, „1“ -enable

<BDAddress>: This is the device address in Classic mode and Access Code in LE mode. It is a 48-bit address in hexadecimal format with colon separation. e.g., 00:23:A7:01:02:03.

<pktlength>: This is the length of the packet, in bytes, to be received. Range is from 1 to 1021.

<pkttype>: This is the type of packet to be received, as per the Bluetooth standard. Range is from 1 to 15.

<linktype>: This parameter indicates the Link Type – ACL, SCO, eSCO. This parameter is valid only in the Classic mode and invalid in LE mode. „0“ – SCO , „1“ – ACL , „2“ – eSCO

<rate>: This parameter decides whether the reception has to happen in Basic Data Rate or Enhanced Data Rate in Classic mode. This parameter is invalid in LE mode. „1“ – Basic Data Rate , „2“ or „3“ – Enhanced Data Rate

<rx_channel>: This parameter indicates the Receive channel index, as per the Bluetooth standard.

<tx_channel>: This parameter indicates the Transmit channel index, as

per the Bluetooth standard.

<scramblerseed>: This parameter is the initial seed to be used for whitening. It should be set to „0“ to disable whitening.

<payloadtype>: This parameter indicates the type of payload to be received.

„0“ – Payload consists of zeros.

„1“ – Payload consists of 0xFF“ s.

„2“ – Payload consists of 0x55“ s

„3“ – Payload consists of 0xF0“ s.

„4“ – Payload consists of PN9 sequence.

<protocolmode>: This parameter is used to choose between Bluetooth Classic and LE modes for the packet reception. „1“ – Classic mode , „2“ – LE Mode

<lechanneltype>: This parameter indicates the channel type in LE mode. „0“ – Advertising channel , „1“ – Data channel

<freqhopenable>: This parameter is used to choose the hopping pattern. „0“ – No hopping , „1“ – Fixed hopping , „2“ – Random hopping

<antennaselect>: This parameter is used to select one of the two RF ports connecting to antennas. For the modules without integrated antenna, it is used to select between pins RF_OUT_1 and RF_OUT_2. For the modules with integrated antenna and U.FL connector, it used to select between the two.
„2“ – RF_OUT_2/Antenna , „3“ – RF_OUT_1/U.FL

2.3.5.3 Per cw mode

Description:

This command mode is used to configure the device to transmit a continuous wave.

Binary Payload Structure

```
typedef struct rsi_bt_cmd_cw_mode {
        UINT08  type;
        UINT08 channel;
        UINT08  cw_mode;
```

```
    UINT08  ant_sel;

} RSI_BT_CMD_CW_MODE;
```

Parameters:

<type>: This parameter indicates that this structure belongs to continuous wave mode. This type has value 13.

<channel>: This parameter indicates the channel index, as per the Bluetooth standard. The range is from 0 to 78.

< cw_mode >: This parameter is used to Start or Stop the Continuous Wave mode transmission. Values are „0“ -start , „2“ - stop.

<ant_sel>: This parameter is used to select one of the two RF ports connecting to antennas. For the modules without integrated antenna, it is used to select between pins RF_OUT_1 and RF_OUT_2. For the modules with integrated antenna and U.FL connector, it used to select between the two. „2“ – RF_OUT_2/Antenna, „3“ – RF_OUT_1/U.FL

2.3.5.4 Per stats

Description:

The following statistics are returned every second.

Binary Payload Structure:

```
typedef struct
    rsi_bt_cmd_per_stats{
    UINT16 crc_fail_count;
    UINT16 crc_pass_count;
    UINT16  RSSI;
    UINT16  Id_pkts;
}RSI_BT_CMD_PER_STATS;
```

Parameters:

Crc_pass_count: The number of packets received which passed CRC

check. Crc_fail_count: The number of packets received which failed CRC

check.

RSSI: The RSSI value of the last received packet. Id_pkts: The number of ID packets received.

2.3.6 Core events

2.3.6.1 Role change status[1]

Description: This event tells about the status of the command "Change Master Slave Role".

Binary Payload Structure:

```
typedef struct rsi_bt_event_role_change_status {

   UINT08 BDAddress[6];

   UINT08 RoleChangeStatus;

} RSI_BT_EVENT_ROLE_CHANGE_STATUS;
```

AT Event Format:

AT+RSIBT_ ROLECHANGESTAT < BDAddress >< RoleChangeStatus >\r\n

Parameters:

BDAddress – Remote BD Address

RoleChangeStatus –

0 – Master role

1 – Slave role

AT event Ex: AT+RSIBT_ ROLECHANGESTAT AA-BB-CC-DD-EE-FF,1\r\n

2.3.6.2 Unbond or Disconnect status

Description:

This event tells about the status of unbonding between the BT module and remote BT device.

Binary Payload Structure:

```
typedef struct rsi_bt_event_unbond_status {

   UINT08 BDAddress[6];
```

[1] This event may not occur since Role Switch is not currently supported.

} RSI_BT_EVENT_UNBOND_STATUS;

AT Event Format:

AT+RSIBT_UNBONDRESP < BDAddress ><unbond_status>\r\n

bond_status :

0 – Success

1 - Failure

Parameters:

BDAddress**:** BD address of the remote BT device in the vicinity of the BT module.

AT event Ex: AT+RSIBT_UNBONDRESP AA-BB-CC-DD-EE-FF,1\r\n

2.3.6.3 Bond Response

Description:

This event will be sent to the host, containing the status for "Bond" command.

Binary Payload Structure:

typedef struct rsi_bt_event_bond_response {

UINT08 BDAddress[6];

} RSI_BT_EVENT_BOND_RESPONSE;

AT Event Format:

AT+RSIBT_BONDRESP < BDAddress ><bond_status><error>\r\n

bond_status :

0 – Success

1 - Failure

Parameters:

BDAddress**:** BD address of the remote BT device in the vicinity of the BT module.

AT event Ex: AT+RSIBT_BONDRESP AA-BB-CC-DD-EE-FF,0\r\n

2.3.6.4 Inquiry response

Description:

This event will be sent in response to the "Inquiry" command. This event contains the details of the BT device (like device name, BD address, COD, RSSI value) in the vicinity of the BT module.

Binary Payload Structure:

```
typedef struct rsi_bt_event_inquiry_response {
   UINT08  InquiryType;
   UINT08  BDAddress[6];
   UINT08  NameLength;
   INT08   RemoteDeviceName[50];
   UINT08  COD[3];
   UINT08  RSSI;
} RSI_BT_EVENT_INQUIRY_RESPONSE;
```

AT Event Format:

AT+RSIBT_INQRESP <InquiryType>,<BDAddress>,<NameLength>,<RemoteDeviceName>\r\n (for Standard Inquiry).

AT+RSIBT_INQRESP

< InquiryType >,< InquiryType NameLength>,<RemoteDeviceName>,<Rssi>,<cod>\r\n (for normal & extended inquiry types with RSSI).

Parameters:

InquiryType**:**

0- Standard Inquiry.

1- Inquiry with RSSI.

2- Extended inquiry with RSSI.

BDAddress**:** BD address of the remote BT device in the vicinity of the BT module. This parameter will be sent by host for all inquiry types.

NameLength**:** Length of the remote device name.

RemoteDeviceName**:** Name of the remote BT device, with corresponding BD address. This parameter will be sent by host for all inquiry types. This parameter will be present only if the **"rmt_name_len"** value is non-zero.

COD**:** Class of the remote BT device. This parameter will be sent by host for all inquiry types.

RSSI**:** RSSI value between the BT module and the remote BT device. This parameter will be sent only for inquiry types 1and 2.

AT event Ex: AT+RSIBT_INQRESP 2,AA-BB-CC-DD-EE-FF,7,redpine,142,7a020c\r\n

2.3.6.5 Remote device name

Description: This event will be sent to the host in response to the "Remote Name Request" command. This event contains the name of the remote BT device, which was requested by the host.

Binary Payload Structure:

```
typedef struct rsi_bt_event_remote_device_name {
   UINT08  BDAddress[6];
   UINT08  NameLength;
   INT08   RemoteDeviceName[50];
} RSI_BT_EVENT_REMOTE_DEVICE_NAME;
```

AT Event Format:

AT+RSIBT_RMTDEVNAME < BDAddress >,< NameLength >,< RemoteDeviceName / ERROR >\r\n

Parameters:

BDAddress**:** BD address of the remote BT device in the vicinity of the BT module.

NameLength**:** Length of the remote device name. If remote device name is not found this parameter will be zero and the **"rmt_dev_name"** contains the error code.

RemoteDeviceName**:** Name of the remote BT device, with corresponding BD address. If the remote device name is not available or remote device is not connected, this parameter contains the corresponding error code.

AT event Ex: AT+RSIBT_RMTDEVNAME AA-BB-CC-DD-EE-FF,7,redpine\r\n

2.3.6.6 Disconnected

Description:

This event is raised when disconnection happens between the local BT device and the remote device.

Binary Payload Structure:

typedef struct rsi_bt_event_disconnected {

UINT08 BDAddress[6];

} RSI_BT_EVENT_DISCONNECTED;

AT Event Format:

AT+RSIBT_DISCONNECTED < BDAddress >,<reason>\r\n
Parameters:

BDAddress – BD address of the remote BT device.

AT event Ex: AT+RSIBT_CLASSIC_DISCONNECTED AA-BB-CC-DD-EE-FF,0\r\n

2.3.6.7 User confirmation Request

Description:

This event is raised when User Confirmation Request comes from the remote BT Device.

Binary Payload Structure:

typedef struct rsi_bt_event_user_confirmation_request {

UINT08 BDAddress[6];

UINT32 ConfirmationValue;

} RSI_BT_EVENT_USER_CONFIRMATION_REQUEST;

AT Event Format:

AT+RSIBT_USRCONFIRMREQ=< BDAddress >,<confirmation_value>\r\n
Parameters:

BDAddress**:** The remote BT device BD address.

confirmationValue**:** Range from 0 to 999999 in decimal.

AT event Ex: AT+RSIBT_USRCONFIRMREQ AA-BB-CC-DD-EE-FF,1234\r\n

2.3.6.8 User passkey display

Description: This event is raised when User Passkey comes from the module.

Binary Payload Structure:

typedef struct rsi_bt_event_user_passkey_display {

UINT08 BDAddress[6];

UINT32 Passkey;

} RSI_BT_EVENT_USER_PASKEY_DISPLAY;

AT Event Format:

AT+RSIBT_USRPASSKEYDISP < BDAddress >,<passkey>\r\n

Parameters:

BDAddress**:** The remote BT device BD address.

passkey**:** Range from 0 to 999999 in decimal.

AT event Ex: AT+RSIBT_USRPASSKEYDISP AA-BB-CC-DD-EE-FF,12345\r\n

2.3.6.9 User pincode request

Description: This event is raised when User Pincode is invoked by the remote BT Device. In such a case, user shall respond with **Pincode Reply** Command.

Binary Payload Structure:

typedef struct rsi_bt_event_user_pincode_request {

UINT08 BDAddress[6];

} RSI_BT_EVENT_USER_PINCODE_REQUEST;

AT Event Format:

AT+RSIBT_USRPINCODEREQ < BDAddress >\r\n

Parameters:

BDAddress – BD Address of the remote BT device.

AT event Ex: AT+RSIBT_USRPINCODEREQ AA-BB-CC-DD-EE-FF\r\n

2.3.6.10 User passkey request

Description: This event is raised when User passkey request comes from the remote BT Device.

Binary Payload Structure:

typedef struct rsi_bt_event_user_passkey_request {

UINT08 BDAddress[6];

} RSI_BT_EVENT_USER_PASSKEY_REQUEST;

AT Event Format:

AT+RSIBT_USRPASSKEYREQ < BDAddress >\r\n

Parameters:

BDAddress – BD Address of the remote BT device.

2.3.6.11 Inquiry complete

Description:

This event is raised after the Inquiry is completed.

Binary Payload Structure:

There is no Payload.

AT Event Format:

AT+RSIBT_INQCOMPLETE\r\n

2.3.6.12 Auth complete

Description: This event tells about the status of authentication process.

Binary Payload Structure:

typedef struct rsi_bt_event_auth_complete {

UINT08 BDAddress[6];

} RSI_BT_EVENT_AUTH_COMPLETE;

AT Event Format:

AT+RSIBT_AUTHENTICATION_STATUS < BDAddress >, <STATUS>, <ERROR>\r\n

Parameters:

BDAddress – BD Address of the remote device.

2.3.6.13 User linkkey Request

Description: This event is raised when user linkkey request comes from the remote BT Device.

Binary Payload Structure:

typedef struct rsi_bt_event_user_linkkey_request {

UINT08 BDAddress[6];

} RSI_BT_EVENT_USER_LINKKEY_REQUEST;

AT Event Format:

AT+RSIBT_USRLNKKEYREQ < BDAddress >\r\n

Parameters:

BDAddress – BD Address of the remote BT device.

AT event Ex: AT+RSIBT_USRLINKKEYREQ AA-BB-CC-DD-EE-FF\r\n

2.3.6.14 User linkkey save

Description: This event is raised when a device is paired and linkkey for remote BT Device is given to host.

Binary Payload Structure:

typedef struct rsi_bt_event_user_linkkey_save {

UINT08 BDAddress[6];

UINT08 LinkKey[16];

} RSI_BT_EVENT_USER_LINKKEY_SAVE;

AT Event Format:

AT+RSIBT_USRLNKKEYSAVE < BDAddress ><LinkKey>\r\n

Parameters:

BDAddress – BD Address of the remote BT device.

LinkKey – Link key for the remote BT device.

AT event Ex: AT+RSIBT_USRLINKKEYSAVE AA-BB-CC-DD-EE-FF,3C,A5,50,25,DC,D0,B0,AB,B7,C3,4F,4D,9,79,2C,5C\r\n

2.3.6.15 SSP Enable

Description: This event is raised when a device pairing mode set to Simple Secure pairing mechanism.

Binary Payload Structure:

typedef struct rsi_bt_event_ssp_complete

{

UINT08 BDAddress[RSI_BT_BD_ADDR_LEN];

UINT08 Status;

} RSI_BT_EVENT_SSP_COMPLETE;

AT Event Format:

AT+RSIBT_SIMPLEPAIRINGCOMPLETED < BDAddress >,<status>\r\n

Parameters:

BDAddress – BD Address of the remote BT device.

Status-

0- Simple Pairing succeeded.

1- Simple pairing failed.

AT event Ex:

AT+RSIBT_SIMPLEPAIRINGCOMPLETED AA-BB-CC-DD-EE-FF,0\r\n

2.3.6.16 Mode change

Description: This event occurs when sniff mode is enable by either remote device or local device and it is raised to indicate whenever the device changes between Active mode and Sniff mode.

Binary Payload Structure:

```
typedef struct rsi_bt_event_mode_change
{
    UINT08  BDAddress[RSI_BT_BD_ADDR_LEN];
    UINT08  CurrentMode;
    UINT08  Reserved;
    UINT16  ModeInterval;
} RSI_BT_EVENT_MODE_CHANGE;
```

AT Event Format:

AT+RSIBT_MODECHANGED < BDAddress >,<CurrentMode>,<Mode Interval>\r\n

Parameters:

BDAddress – BD Address of the remote BT device.

CurrentMode - State the connection is currently in.

ModeInterval - Specify a time amount specific to each state.

AT event Ex:

AT+RSIBT_MODECHANGED AA-BB-CC-DD-EE-FF,1,192\r\n

2.3.6.17 Sniff subrating[1]

Description: This event is raised to indicate that the device has either enabled sniff subrating or the sniff subrating parameters have been renegotiated by the link manager.

Binary Payload Structure:

```
typedef struct rsi_bt_event_sniff_subrating
{
        UINT08  BDAddress[RSI_BT_BD_ADDR_LEN];
        UINT16  MaxTxLatency;
        UINT16  MinRemoteTimeout;
        UINT16  MinLocalTimeout;
} RSI_BT_EVENT_SNIFF_SUBRATING;
```

AT Event Format:

AT+RSIBT_SNIFFSUBRATING < BDAddress >,<MaxTxlatency>,
< MinRemoteTimeout >,< MinLocalTimeout >\r\n

Parameters:

BDAddress – BD Address of the remote BT device.

MaxTxLatency- Maximum latency for data being transmitted from the local device to the remote device.

MinRemoteTimeout – The base sniff subrate timeout in baseband slots that the remote device shall use.

MinLocalTimeout- The base sniff subrate timeout in baseband slots that the local device will use.

AT event Ex:

AT+RSIBT_SNIFFSUBRATING AA-BB-CC-DD-EE-FF,192,1000,1000\r\n

2.3.7 SPP events

2.3.7.1 SPP Receive

Description: This event is sent to the host when data is received from remote BT device through SPP. This data_len field contains the length of data to be send to the host.

Binary Payload Structure:

```
typedef struct  {
```

[1] Currently, Sniff subrating mode is not supported.

```
  UINT16  DataLength;

  UINT08  Data[200];

} RSI_BT_EVENT_SPP_RECEIVE;
```

AT Event Format:

AT+RSIBT_SPPRX < DataLength >< Data >\r\n

Parameters: Datalength – length of the data (from 1 byte to 200 bytes).

AT event Ex: AT+RSIBT_SPPRX 4,rtrt\r\n

2.3.7.2 SPP connected

Description: This event will be sent to the host, when the connection is established between BT Module and the remote BT device based on SPP profile.

Binary Payload Structure:

```
typedef struct rsi_bt_event_spp_connected {

  UINT08  BDAddress[6];

} RSI_BT_EVENT_SPP_CONNECTED;
```

AT Event Format:

AT+RSIBT_SPPCONNECTED < BDAddress >\r\n

Parameters:

BDAddress – BD Address of the remote BT device.

AT event Ex: AT+RSIBT_SPPCONNECTED AA-BB-CC-DD-EE-FF\r\n

2.3.7.3 SPP Disconnected

Description: This event will be sent to the host, when the existing SPP connection of the BT Module with the remote BT device is disconnected.

Binary Payload Structure:

```
typedef struct rsi_bt_event_spp_disconnected {

  UINT08  BDAddress[6];

} RSI_BT_EVENT_SPP_DISCONNECTED;
```

AT Event Format:

AT+RSIBT_SPPDISCONNECTED < BDAddress >\r\n

Parameters:

BDAddress – BD Address of the disconnected device

AT event Ex: AT+RSIBT_SPPDISCONNECTED AA-BB-CC-DD-EE-FF\r\n

2.3.8 Apple IAP1 Events

2.3.8.1 IAP Connected

Description: This event indicates that the connection was established between an Apple device and the accessory.

AT Event Format:

AT+RSIBT_IAP_CONNECTED < BDAddress >,<IAP_Protocol_Ver>\r\n

Parameters:

bd_addr: BD address of the remote BT device.

IAP_Protocol_Version: Version of the IAP supported by the Apple device, which was connected. 1 for IAP1 and 2 for IAP2.

AT event Ex:

AT+RSIBT_IAP_CONNECTED AA-BB-CC-DD-EE-FF,1\r\n

2.3.8.2 IAP Disconnected

Description: This event will be sent to the host, when the connection between the accessory and an Apple device get disconnected.

AT Event Format:

AT+RSIBT_IAP_DISCONNECTED bd_addr>,<IAP_Protocol_Ver>\r\n

Parameters:

bd_addr: BD address of the remote BT device.

IAP_Protocol_Version: Version of the IAP supported by the Apple device, which was connected. 1 for IAP1 and 2 for IAP2.

AT event Ex:

AT+RSIBT_IAP_DISCONNECTED AA-BB-CC-DD-EE-FF,1\r\n

2.3.8.3 IAP1 Accessory Authentication Started

Description: This indicates the start of the Accessory Authentication procedure by the Apple device.

AT Event Format:

AT+RSIBT_IAP1_ACCESSORY_AUTH_STARTED\r\n

Parameters:

None

AT event Ex:

AT+RSIBT_IAP1_ACCESSORY_AUTH_STARTED\r\n

2.3.8.4 IAP1 Accessory Authentication Failed

Description: When the Accessory Authentication gets failed, it will be indicated by this event. The reason for failure is indicated by the error code.

AT Event Format:

AT+RSIBT_IAP1_ACCESSORY_AUTH_FAILED <error_code>\r\n

Parameters:

error_code: The reason for failure is indicated by this error code.

AT event Ex:

AT+RSIBT_IAP1_ACCESSORY_AUTH_FAILED 0x8002\r\n

2.3.8.5 IAP1 Accessory Authentication Completed

Description: This event indicates the completion of the Authentication procedure by the Apple device.

AT Event Format:

AT+RSIBT_IAP1_ACCESSORY_AUTH_COMPLETED\r\n

Parameters: None

AT event Ex:

AT+RSIBT_IAP1_ACCESSORY_AUTH_COMPLETED\r\n

2.3.8.6 IAP1 Now Playing Application Bundle Name

Description: This event contains the current playing application bundle name. This event will be generated when the music player starts playing, when the voice memo starts recording etc.

AT Event Format:

AT+RSIBT_IAP1_CURR_APP_BUNDLE_NAME <name>\r\n

Parameters:

name: Name of the current playing application name.

AT event Ex:

AT+RSIBT_IAP1_CURR_APP_BUNDLE_NAME com.apple.mobileipod\r\n

2.3.8.7 IAP1 Now Playing Application Display Name

Description: This event contains the current playing application Display name. This event will be generated when the music player starts playing, when the voice memo starts recording etc.

AT Event Format:

AT+RSIBT_IAP1_CURR_APP_DISPLAY_NAME <name>\r\n

Parameters:

name: Name of the current playing application name.

AT event Ex:

AT+RSIBT_IAP1_CURR_APP_DISPLAY_NAME Music\r\n

2.3.8.8 IAP1 Assistive Touch Status

Description: This event indicates a change in the status of the Assistive Touch. This event will be sent to the host, whenever there is a change (ON/OFF) in the status of Assistive Touch.

AT Event Format:

AT+RSIBT_IAP1_ASSISTIVE_TOUCH <status>\r\n

Parameters:

status: Status of Assistive touch. 1 for ON and 0 for OFF.

AT event Ex:

AT+RSIBT_IAP1_ASSISTIVE_TOUCH 1\r\n

2.3.8.9 IAP1 IPodOut Status

Description: This event denotes the status of Ipod out.

AT Event Format:

AT+RSIBT_IAP1_IPODOUT <status>\r\n

Parameters:

status: Status of ipod out. **1** for ON and **0** for OFF.

AT event Ex:

AT+RSIBT_ IAP1IPODOUT 0\r\n

2.3.8.10 IAP1 Flow Control Status

Description: This event will be sent to the host, when the connection is established between BT Module and the remote BT device based on SPP profile.

AT Event Format:

AT+RSIBT_IAP1_FLOW_CONTROL <IAP_Protocol_Ver>\r\n

Parameters:

IAP_Protocol_Version: Version of the IAP supported by the Apple device, which was connected. 1 for IAP1 and 2 for IAP2.

AT event Ex:

AT+RSIBT_IAP1_FLOW_CONTROL\r\n

2.3.8.11 IAP1 Radio Tagging Status

Description: This event indicates the radio tagging status.

AT Event Format:

AT+RSIBT_IAP1_RADIO_TAGGING <status>\r\n

Parameters:

Status	Description
0	Tagging operation successful
1	Tagging operation failed
2	Information available for tagging(all required RT)
3	Information not available for tagging

AT Event Ex:

AT+RSIBT_IAP1_RADIO_TAGGING 0\r\n

2.3.8.12 IAP1 Camera status

Description: This event indicates the camera status.

AT Event Format:

AT+RSIBT_IAP1_CAMERA <status>\r\n

Parameters:

Status	Description
0	Camera App off
3	Preview
4	Recording

AT Event Ex:

AT+RSIBT_IAP1_CAMERA 3\r\n

2.3.8.13 IAP1 Database changed status

Description: This event will be raised when there is a change in the Apple device database.

AT Event Format:

AT+RSIBT_IAP1_DATABASE_CHANGED\r\n

Parameters:

None

AT Event Ex:

AT+RSIBT_IAP1_DATABASE_CHANGED\r\n

2.3.8.14 IAP1 Session Space Available Notification

Description: This event will be sent to the host, when the connection is established between BT Module and the remote BT device based on SPP profile.

AT Event Format:

AT+RSIBT_IAP1_SESSION_SPACE_AVAILABLE\r\n

Parameters:

None

AT Event Ex:

AT+RSIBT_IAP1_SESSION_SPACE_AVAILBLE\r\n

2.3.8.15 IAP1 Bluetooth Status

Description: This event will be sent to the host, when the connection is established between BT Module and the remote BT device based on SPP profile.

AT Event Format:

AT+RSIBT_IAP1_BLUETOOTH_STATUS <IAP_Protocol_Ver>\r\n

Parameters :

IAP_Protocol_Version: Version of the IAP supported by the Apple device, which was connected. 1 for IAP1 and 2 for IAP2.

AT Event Ex:

AT+RSIBT_IAP1_BLUETOOTH_STATUS\r\n

2.3.8.16 IAP1 Voiceover Parameter Changed status

Description: This event occurs whenever there is a change in any of the voiceover parameter like voiceover volume, speech rate.

AT Event Format:

AT+RSIBT_IAP1_VOICEOVER_PARAM_CHANGED
<param_type>,<changed_value>\r\n

Parameters:

param_type: Type of the parameter changed.

0 – Voiceover Volume.

1 – Voiceover speech rate.

changed_value: The changed value of the voiceover parameter.

AT Event Ex:

AT+RSIBT_IAP1_VOICEOVER_PARAM_CHANGED 0,150\r\n

2.3.8.17 IAP1 Application Data Session Opened

Description: This event indicates that a session is opened by an Apple device. The details of the session was denoted by the session id and protocol index, which was assigned through the EAProtocoToken at the time of Identification Procedure. If the data session is accepted by the user, this must be acknowledgment by **"”** command.

AT Event Format:

AT+RSIBT_IAP1_OPEN_DATA_SESSION? <session_id>,<protocol_index>\r\n

Parameters:

session_id: ID of the session opened by the Apple device.

protocol_index: Protocol index assigned for an application at the time of Identification.

AT Event Ex:

AT+RSIBT_IAP1_OPEN_DATA_SESSION? 25,1\r\n

2.3.8.18 IAP1 Application Data Session Closed

Description: This event indicates the closure of the data session by the Apple device. This must be acknowledged by **"”** command.

AT Event Format:

AT+RSIBT_IAP1_CLOSE_DATA_SESSION? <session_id>\r\n

Parameters:

session_id: Session ID of the application to be closed.

AT Event Ex:

AT+RSIBT_IAP1_CLOSE_DATA_SESSION? 25\r\n

2.3.8.19 IAP1 Ipod Data Received

Description: This event indicates the data received from the Apple device. If the data is received correctly, an acknowledgment is sent to the Apple device.

AT Event Format:

AT+RSIBT_IAP1_IPOD_DATA_RECEIVED

<session_id>,<data_len>,<data>\r\n

Parameters:

session_id: Session ID

data_len: Length of the data received from the Apple device.

data: Actual data received from the Apple device.

AT Event Ex:

AT+RSIBT_IAP1_IPOD_DATA_RECEIVED 25,7,welcome\r\n

2.3.8.20 IAP1 Accessory HID Report

Description: This event indicates the accessory HID report sent by the Apple device. For example an LED report sent to the keyboard accessory.

AT Event Format:

AT+RSIBT_IAP1_IPOD_DATA_RECEIVED
<desc_index>,<report_type>,<len>,<report>\r\n

Parameters:

desc_index: HID descriptor index, already registered with the Apple device.

report_type**:** Input report.

len: Length of the report in bytes.

report: Device report. The report values are not the ASCII values, but the numerical values.

AT Event Ex:

AT+RSIBT_IAP1_ACCESSORY_HID_REPORT 0,1,3, 01 00 00 \r\n

3 Bluetooth Classic Error Codes

3.1.1 Generic Error Codes

Error Code	Description
0x4E01	Unknown HCI command
0x4E02	Unknown Connection Identifier
0x4E03	Hardware failure
0x4E04	Page timeout
0x4E05	Authentication failure
0x4E06	Pin missing
0x4E07	Memory capacity exceeded
0x4E08	Connection timeout
0x4E09	Connection limit exceeded
0x4E0A	SCO limit exceeded
0x4E0B	ACL Connection already exists
0x4E0C	Command disallowed
0x4E0D	Connection rejected due to limited resources
0x4E0E	Connection rejected due to security reasons
0x4E0F	Connection rejected for BD address
0x4E10	Connection accept timeout
0x4E11	Unsupported feature or parameter
0x4E12	Invalid HCI command parameter
0x4E13	Remote user terminated connection
0x4E14	Remote device terminated connection due to low resources
0x4E15	Remote device terminated connection due to power off
0x4E16	Local device terminated connection
0x4E17	Repeated attempts
0x4E18	Pairing not allowed
0x4E19	Unknown LMP PDU

0x4E1A	Unsupported remote feature
0x4E1B	SCO offset rejected
0x4E1C	SCO interval rejected
0x4E1D	SCO Air mode rejected
0x4E1E	Invalid LMP parameters
0x4E1F	Unspecified
0x4E20	Unsupported LMP Parameter
0x4E21	Role change not allowed
0x4E22	LMP response timeout
0x4E23	LMP transaction collision
0x4E24	LMP PDU not allowed
0x4E25	Encryption mode not acceptable
0x4E26	Link key cannot change
0x4E27	Requested QOS not supported
0x4E28	Instant passed
0x4E29	Pairing with unit key not supported
0x4E2A	Different transaction collision
0x4E2B	Reserved 1
0x4E2C	QOS parameter not acceptable
0x4E2D	QOS rejected
0x4E2E	Channel classification not supported
0x4E2F	Insufficient security
0x4E30	Parameter out of mandatory range
0x4E31	Reserved 2
0x4E32	Role switch pending
0x4E33	Reserved 3
0x4E34	Reserved slot violation
0x4E35	Role switch failed
0x4E36	Extended Inquiry Response too large
0x4E37	Extended SSP not supported

0X4E38	Host busy pairing
0x4E39	Wrong BD Address

Table 5 Bluetooth Generic Error Codes

3.1.2 Core Error Codes

Error Code	Description
0x4040	IO Fail
0x4041	Unknown
0x4042	HW Busy
0x4043	Max Sock
0x4044	Short Buf
0x4045	Max Name Size
0x4046	Invalid Args
0x4047	Sock open fail
0x4048	Timeout
0x4049	Socket state invalid
0x404A	Bad bd address
0x404B	Acl packet error
0x404C	Pool alloc fail
0x404D	Tx fail
0x404E	Connection refused
0x404F	Confirmation result
0x4050	Remote user disconnected
0x4051	Remote device not responding
0x4052	Invalid command
0x4053	Unsupported feature param value
0x4054	Thread create fail
0x4055	Sem wait fail
0x4056	Pool full
0x4057	Hw buffer overflow

0x4058	Tx buffer empty
0x4059	HCI connection fail
0x405A	Operation incomplete
0x405B	Operation cancel
0x405C	BSP error
0x4060	Sco connection fail
0x4061	No HCI connection
0x4062	Socket disconnected
0x4063	Socket timeout
0x4064	HCI connection encrypt fail
0x4065	Max acl packet buffer length
0x4066	Max nbr acl packets
0x4067	Invalid state
0x4069	Remote name fail
0x406A	Invalid response
0x4071	Invalid psm
0x4072	Psm in use
0x4073	Invalid hci connection handle
0x4074	Invalid cid
0x4075	Invalid pkt
0x4080	Scn is in use
0x4081	Max acl connections
0x4082	Sock already exists
0x4100	Invalid pdu
0x4101	Invalid pdu data element
0x4102	Sdp service not found
0x4103	Sdp attribute not found
0x4104	Sdp max service attribute
0x4200	Max RF communication channels

0x4201	RF communication disconnected
0x4202	RF communication channel not found
0x4203	RF communication invalid packet
0x4204	RF communication remote credits zero
0x4205	RF communication invalid state
0x4206	RF communication fcoff
0x4207	RF communication no service connection
0x4300	HCI connection already exists
0x4301	Max hci connection
0x4302	SCO invalid state
0x0102	Unknown
0x0103	Timeout
0x0104	Memory alloc fail
0x0106	Io fail
0x0108	Unsupported
0x0109	Short buf
0x010A	Buf overflow
0x010B	Too large buf
0x010C	Io abort
0x010D	File open fail
0x1010	Os task invalid prio
0x1011	Os task prio exists
0x1012	Os task not stopped
0x1020	Os sem max value
0x1021	Os sem not available
0x1022	Os sem reset
0x1030	Os mutex not owner
0x1031	Os mutex not locked
0x1032	Os mutex lock failed

0x1033	Os mutex try lock failed
0x1040	Os msg queue full
0x1041	Os message queue empty
0x1050	Pipe empty
0x1051	Pipe full
0x1052	Invalid len
0x1053	Pipe read in use
0x1054	Pipe write in use
0x1060	Os timer expired
0x1061	Os timer state running
0x1070	Os can not wait
0x1080	Os mem pool empty
0x1081	Os mem pool size short
0x4500	SPP not connected
0x4501	SPP not initialized
0x4FF9	Inquiry cancel command is given when device is not in Inquiry State
0x4604	SPP Tx FAIL

Table 6 BT Classic Error Codes

Error Code	Description
0x1090	Os Event queue full
0x1091	Os Event not available
0x1092	Os Event not created
0x1093	Os Event prio not created
0x1094	Os Event no event created

Table 7 BT Event Queue Error Codes

Error Code	Description
0x8000	ERR_IAP1_SUCCESS
0x8001	ERR_IAP1_UNKNOWN_DATABASE
0x8002	ERR_IAP1_COMMAND_FAILED
0x8003	ERR_IAP1_DEVICE_OUTOF_RESOURCE
0x8004	ERR_IAP1_BAD_PARAM
0x8005	ERR_IAP1_UNKNOWN_ID
0x8006	ERR_IAP1_COMMAND_PENDING
0x8007	ERR_IAP1_NOT_AUTHENTICATED
0x8008	ERR_IAP1_BAD_AUTHENTICATION_VERSION
0x8009	ERR_IAP1_ACCESSORY_PWR_REQ_FAILED
0x800A	ERR_IAP1_CERTIFICATE_INVALID
0x800B	ERR_IAP1_CERTIFICATE_PERMISSION_FAILED
0x800C	ERR_IAP1_FILE_IN_USE
0x800D	ERR_IAP1_INVALID_FILE_HANDLE
0x800E	ERR_IAP1_DIRECTORY_NOT_EMPTY
0x800F	ERR_IAP1_OPERATION_TIMED_OUT
0x8010	ERR_IAP1_COMMAND_UNAVAILABLE
0x8011	ERR_IAP1_ACC_DETECT_NOT_GROUNDED
0x8012	ERR_IAP1_SELECTION_NOT_GENIUS
0x8013	ERR_IAP1_MULTISECTION_DATA_RECV_SUCCESS
0x8014	ERR_IAP1_LINGO_BUSY
0x8015	ERR_IAP1_MAX_ACCESSORY_CONN_REACHED
0x8016	ERR_IAP1_HID_DESC_INDEX_INUSE
0x8017	ERR_IAP1_DROPPED_DATA

0x8018	ERR_IAP1_UNSUPPORTED_IPODOUT_VIDEO_SETTINGS
0x8301	ERR_IAP_CP_SUCCESS
0x8302	ERR_IAP_CP_INVALID_READ_REGISTER
0x8303	ERR_IAP_CP_INVALID_WRITE_REGISTER
0x8304	ERR_IAP_CP_INVALID_SIGNATURE_LEN
0x8305	ERR_IAP_CP_INVALID_CHALLENGE_LEN
0x8306	ERR_IAP_CP_INVALID_CERTIFICATE_LEN
0x8307	ERR_IAP_CP_SIGNATURE_GENERATION
0x8308	ERR_IAP_CP_CHALLENGE_GENERATION
0x8309	ERR_IAP_CP_SIGNATURE_VERIFICATION
0x830A	ERR_IAP_CP_CERITIFICATE_VERIFICATION
0x830B	ERR_IAP_CP_INVALID_PROCESS_CTRL
0x830C	ERR_IAP_CP_PROCESS_CTRL_OUTOF_SEQUENCE
0x4FF8	ERR_INVALID_COMMAND
0x8100	ERR_IAP1_IDPS_SUCCESS
0x8101	ERR_IAP1_IDPS_TKN_FIELDS_REJECTED
0x8102	ERR_IAP1_IDPS_TKN_FIELDS_MISSING
0x8103	ERR_IAP1_IDPS_TKN_FIELDS_INCORRECT_RESEND
0x8104	ERR_IAP1_IDPS_ACCESSORY_MAY_RETRY
0x8105	ERR_IAP1_IDPS_TIMEOUT
0x8106	ERR_IAP1_IDPS_NOT_SUPPORTED
0x8107	ERR_IAP1_IDPS_INVALID_TKN_FIELDS

Table 8 BT IAP Error Codes

4 Bluetooth API Library

4.1 API File Organization

Bluetooth APIs are organized into following directory structure

	Path(Within *RS9113.WC.GENR.x.x.x* folder)
BT APIs	host/apis/bt/core/
Interface Specific APIs	host/apis/intf/
HAL APIs	host/apis/hal/
BT Reference Applications	host/apis/bt/ref_apps/
BT Linux Application	RS9113.WC.GENR.xxx/host/reference_projects/ LINUX/Application/bt/src
Linux USB Driver	host/reference_projects/LINUX/Driver/usb/src

4.2 API Prototypes

4.2.1 Generic Prototypes

4.2.1.1 Set Local name

INT16 rsi_bt_set_local_name(RSI_BT_CMD_SET_LOCAL_NAME *SetLocalName);

4.2.1.2 Query Local name

INT16 rsi_bt_query_local_name(void);

4.2.1.3 Set Local COD

INT16 rsi_bt_set_local_cod(RSI_BT_CMD_SET_LOCAL_COD *SetLocalCOD);

4.2.1.4 Query Local COD

INT16 rsi_bt_query_local_cod(void);

4.2.1.5 Query RSSI

INT16 rsi_bt_query_rssi(RSI_BT_CMD_QUERY_RSSI *GetRSSI);

4.2.1.6 Query Link Quality

INT16 rsi_bt_query_link_quality(RSI_BT_CMD_QUERY_LINK_QUALITY *GetLinkQuality);

4.2.1.7 Query Local BD Address

INT16 rsi_bt_query_local_bd_address(void);

4.2.1.8 Initialize BT Module

INT16 rsi_bt_device_init(void);

4.2.1.9 Deinitialize BT Module

```
INT16 rsi_bt_device_deinit(void);
```

4.2.1.10 BT Antenna Select

```
INT16 rsi_bt_antenna_select(RSI_BT_CMD_ANTENNA_SELECT
*uAntennaSelect);
```

4.2.2 BT Classic Core Prototypes

4.2.2.1 Set Profile mode

```
INT16 rsi_bt_set_profle_mode(RSI_BT_CMD_SET_PROFILE_MODE   *SetProfMode );
```

4.2.2.2 Set Discovery mode

```
INT16 rsi_bt_set_discover_mode(RSI_BT_CMD_SET_DISCV_MODE *SetDiscvMode);
```

4.2.2.3 Get Discovery mode

```
INT16 rsi_bt_get_discover_mode(void);
```

4.2.2.4 Set Connectability mode

```
INT16 rsi_bt_set_conn_mode(RSI_BT_CMD_SET_CONN_MODE *SetConnMode);
```

4.2.2.5 Get Connectability mode

```
INT16 rsi_bt_get_conn_mode(void);
```

4.2.2.6 Set Pair Mode

```
INT16 rsi_bt_set_pair_mode(RSI_BT_CMD_SET_PAIR_MODE
*SetPairMode);
```

4.2.2.7 Get Pair Mode

```
INT16 rsi_bt_query_pair_mode(void);
```

4.2.2.8 Remote Name Request

```
INT16
rsi_bt_remote_name_request(RSI_BT_CMD_REMOTE_NAME_REQUEST
*RemNameReq);
```

4.2.2.9 Remote Name Request Cancel

```
INT16
rsi_bt_remote_name_request_cancel(RSI_BT_CMD_REMOTE_NAME_R
EQUEST_CANCEL *RemNameReqCancel);
```

4.2.2.10 Inquiry

```
INT16 rsi_bt_inquiry(RSI_BT_CMD_INQUIRY *Inq);
```

4.2.2.11 Inquiry Cancel

```
INT16 rsi_bt_inquiry_cancel(void);
```

4.2.2.12 Bond or Create Connection

INT16 rsi_bt_bond(RSI_BT_CMD_BOND *Bond);

4.2.2.13 Bond Cancel or Create Connection Cancel

INT16 rsi_bt_bond_cancel(RSI_BT_CMD_BOND_CANCEL *BondCancel);

4.2.2.14 Unbond or Disconnect

INT16 rsi_bt_unbond(RSI_BT_CMD_UNBOND *Unbond);

4.2.2.15 Set Pin Type

INT16 rsi_bt_set_pin_type(RSI_BT_CMD_SET_PIN_TYPE *SetPinType);

4.2.2.16 Get Pin Type

INT16 rsi_bt_query_pin_type(void);

4.2.2.17 User Confirmation

INT16 rsi_bt_user_confirmation(RSI_BT_CMD_USER_CONFIRMATION *UserConf);

4.2.2.18 Passkey Request Reply

INT16 rsi_bt_passkey_request_reply(RSI_BT_CMD_PASSKEY_REPLY *PasKeyReply);

4.2.2.19 Pincode Reply

INT16 rsi_bt_pincode_reply(RSI_BT_CMD_PINCODE_REPLY *PincodeReply);

4.2.2.20 Get Local Device Role

INT16 rsi_bt_query_role(RSI_BT_CMD_QUERY_ROLE *GetRole);

4.2.2.21 Set Local Device Role

INT16 rsi_bt_set_role(RSI_BT_CMD_SET_ROLE *SetRole);

4.2.2.22 Get Service List

INT16 rsi_bt_query_services(RSI_BT_CMD_QUERY_SERVICES *GetServ);

4.2.2.23 Search Service

INT16 rsi_bt_search_service(RSI_BT_CMD_SEARCH_SERVICE *SearchServ);

4.2.2.24 Linkkey Reply

INT16 rsi_bt_linkkey_reply(RSI_BT_CMD_LINKKEY_REPLY *LinkKeyReply);

4.2.2.25 Enable SSP mode

INT16 rsi_bt_enable_ssp_mode(RSI_BT_CMD_SET_SSP_MODE *Setsspmode);

4.2.2.26 Accept SSP confirm

INT16 rsi_bt_accept_ssp_confirm(RSI_BT_CMD_ACCEPT_SSP_CONFIRM *acceptssp);

4.2.2.27 Reject SSP confirm

INT16 rsi_bt_reject_ssp_confirm(RSI_BT_CMD_REJEcT_SSP_CONFIRM *rejectssp);

4.2.2.28 Start sniff mode

INT16 rsi_bt_start_sniff_mode (RSI_BT_CMD_SNIFF_MODE *SniffMode);

4.2.2.29 Exit sniff mode

INT16 rsi_bt_exit_sniff_mode (RSI_BT_CMD_SNIFF_EXIT *SniffExit);

4.2.2.30 Sniff subrating mode[1]

INT16 rsi_bt_sniff_subrating_mode (RSI_BT_CMD_SNIFF_SUBRATING *SniffSubrating);

4.2.3 BT SPP Prototypes

4.2.3.1 SPP connect

INT16 rsi_bt_spp_connect(RSI_BT_CMD_SPP_CONNECT *SPPConn);

4.2.3.2 SPP Disconnect

INT16 rsi_bt_spp_disconnect(RSI_BT_CMD_SPP_DISCONNECT *SPPDisConn);

4.2.3.3 SPP Transfer

INT16 rsi_bt_spp_transfer(RSI_BT_CMD_SPP_TRANSFER *SPPTransfer);

[1] Currently, Sniff Subrating mode is not supported.

5 Application

The files in the Applications folders contain files for the application layer of the Host MCU. These have to be modified to setup the application for the system which the user wants to realize. The user has to call the APIs provided in the API library to configure the BT device.

1. bt_main.c – This file contains the entry point for the application. It also has the initialization of parameters of the global structure and the operations to control & configure the module, like inquiry, scan etc. Here we just provided sample code for the user to understand the flow of commands. This is not must to use the same. User can write his own application code instead of that.
2. rsi_app_util.h and rsi_app_util.c – These files contain list of utility functions which are used by rsi_bt_config_init API and debug prints.
3. rsi_bt_config.h and rsi_bt_config_init.c – These files contain all the parameters to configure the module. Some example parameters are BD Address of the device with which the module should connect, etc.

To facilitate Application development we have defined a data structure named RSI_BT_API as described below.This structure is initialized by the application using rsi_bt_init_struct API of the rsi_bt_config_init.c file (application layer file). The user may change the values assigned to the macros without worrying about understanding the contents of the structure.

The contents of this structure are explained in brief below, using the declaration of the structure in rsi_bt_global.h file (which is also an application layer file and is placed in core APIs include folder).

```
typedef union{
  RSI_BT_CMD_SET_LOCAL_NAME            uSetLocalName;
  RSI_BT_CMD_SET_LOCAL_COD             uSetLocalCOD;
  RSI_BT_CMD_QUERY_RSSI               uQryRssi;
  RSI_BT_CMD_QUERY_LINK_QUALITY         uQryLinkQuality;
  RSI_BT_CMD_ANTENNA_SELECT            uAntennaSelect;
  RSI_BT_CMD_SET_PROFILE_MODE           uSetProfMode;
  RSI_BT_CMD_SET_DISCV_MODE            uSetDiscvMode;
  RSI_BT_CMD_SET_CONN_MODE             uSetConnMode;
  RSI_BT_CMD_SET_PAIR_MODE             uSetPairMode;
  RSI_BT_CMD_REMOTE_NAME_REQUEST        uRemNameReq;
  RSI_BT_CMD_REMOTE_NAME_REQUEST_CANCEL
uRemNameReqCancel;
  RSI_BT_CMD_INQUIRY                  uInq;
```

```
 RSI_BT_CMD_BOND                 uBond;
 RSI_BT_CMD_BOND_CANCEL          uBondCancel;
 RSI_BT_CMD_UNBOND               uUnbond;
 RSI_BT_CMD_SET_PIN_TYPE         uSetPinType;
 RSI_BT_CMD_USER_CONFIRMATION    uUserConf;
 RSI_BT_CMD_PASSKEY_REPLY        uPasskeyReply;
 RSI_BT_CMD_PINCODE_REPLY        uPincodeReply;
 RSI_BT_CMD_QUERY_ROLE           uGetRole;
 RSI_BT_CMD_SET_ROLE             uSetRole;
 RSI_BT_CMD_QUERY_SERVICES       uGetServ;
 RSI_BT_CMD_SEARCH_SERVICE       uSearchServ;
 RSI_BT_CMD_LINKKEY_REPLY        uLinkKeyReply;
 RSI_BT_CMD_SET_SSP_MODE         uSspmodeBuf;
 RSI_BT_CMD_ANTENNA_SELECT *uAntennaSelect;
RSI_BT_CMD_SPP_CONNECT           uSPPConn;
RSI_BT_CMD_SPP_DISCONNECT        uSPPDisConn;
RSI_BT_CMD_SPP_TRANSFER          uSPPTransfer;
RSI_BT_CMD_SNIFF_MODE            uSniffEnable;
RSI_BT_CMD_SNIFF_EXIT            uSniffDisable;
RSI_BT_CMD_SNIFF_SUBRATING       uSniffSubrating;
RSI_BT_CMD_PER_CW_MODE           uPerCw;
RSI_BT_CMD_PER_TRANSMIT          uPerTransmit;
RSI_BT_CMD_PER_RECEIVE           uPerReceive;
RSI_BT_CMD_PER_STATS             uPerStats;

}RSI_BT_API;
```

6 Appendix A: Sample flow

6.1 Configure BT device in Master mode

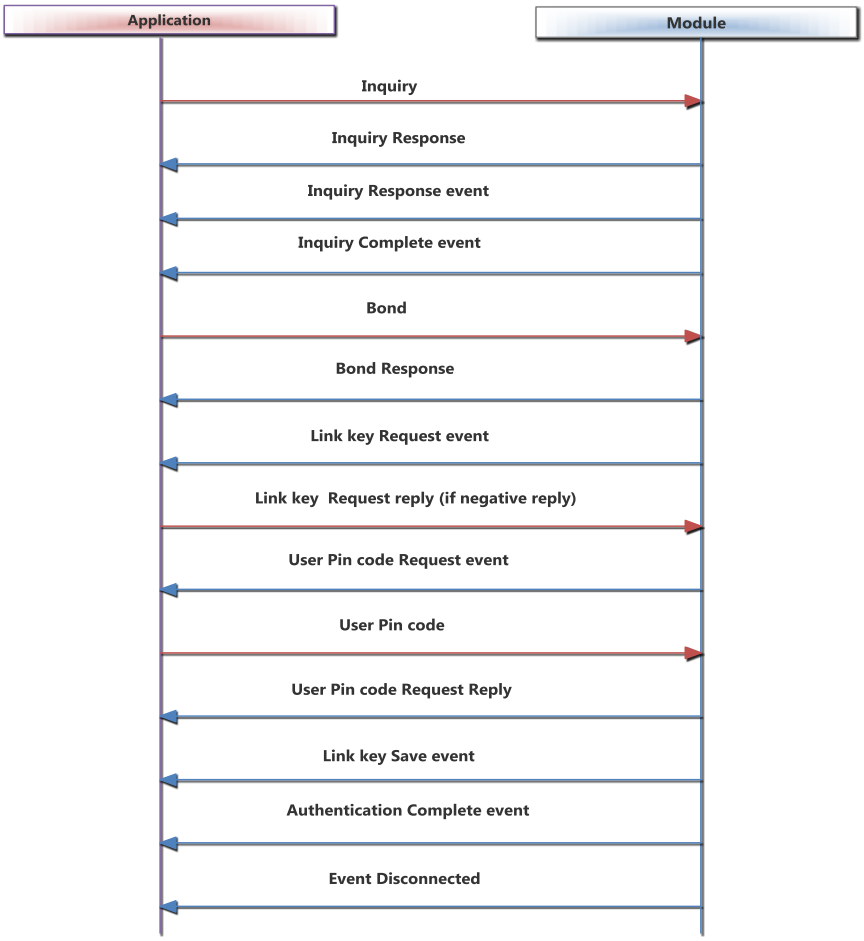

Figure 3 Sample flow in BT master Mode while Link key reply is negative

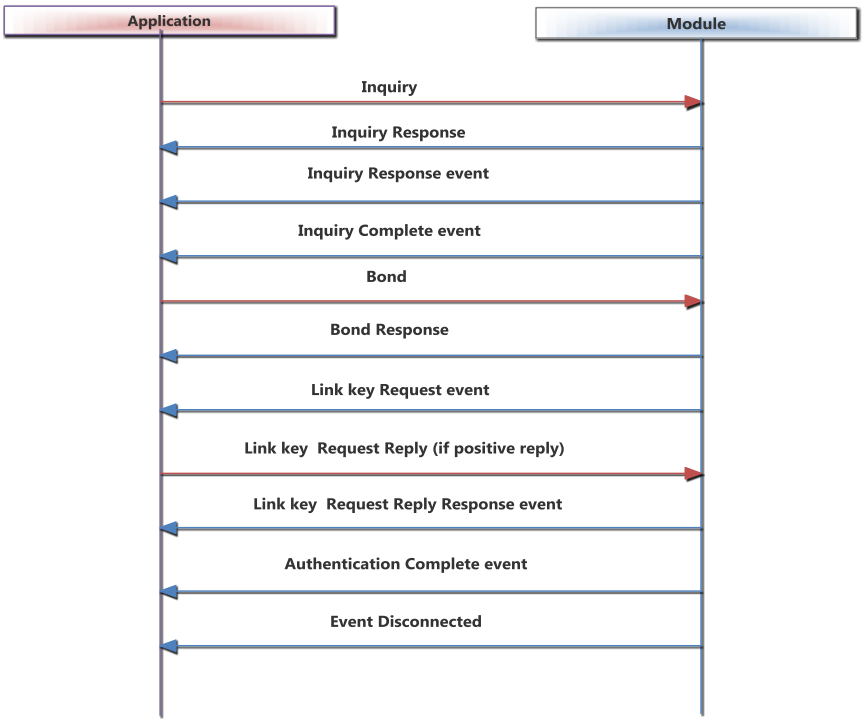

Figure 4 Sample flow in BT master Mode while Link key reply is positive

6.2 Configure BT device in Slave mode

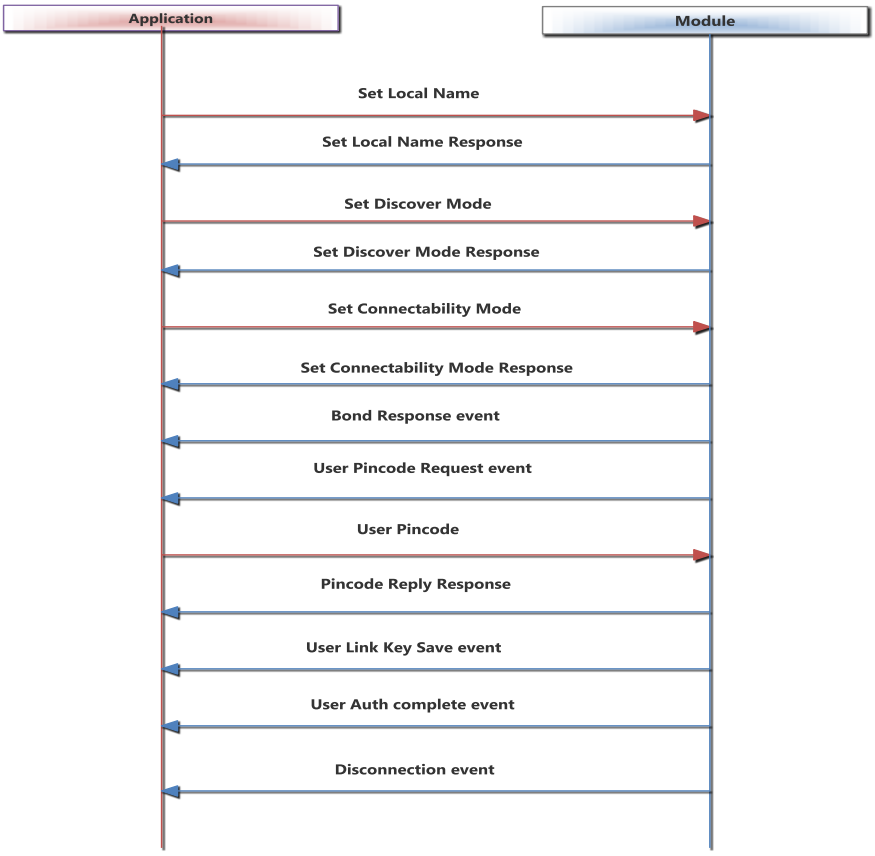

Figure 5 Sample flow in BT Slave Mode

6.3 Configure BT device in Master Mode and do SPP Tx

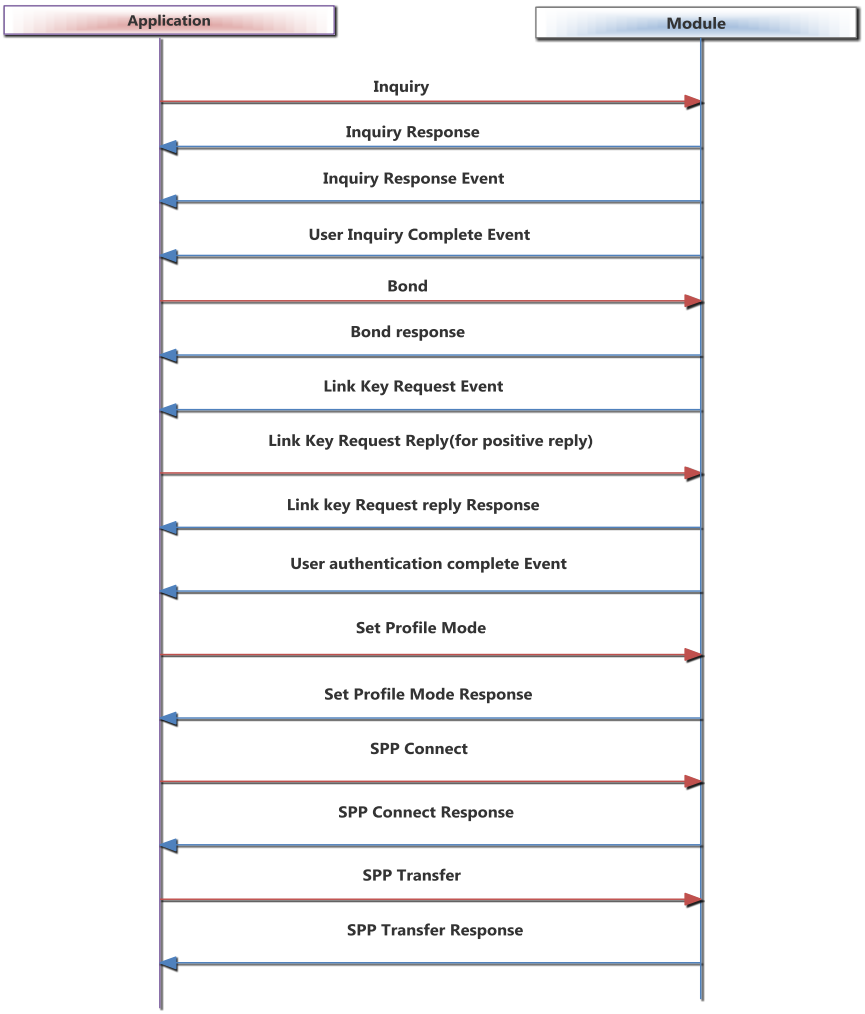

Figure 6 Sample flow in BT Master Mode and do SPP Tx

6.4 Configure BT device in Slave Mode and do SPP Tx

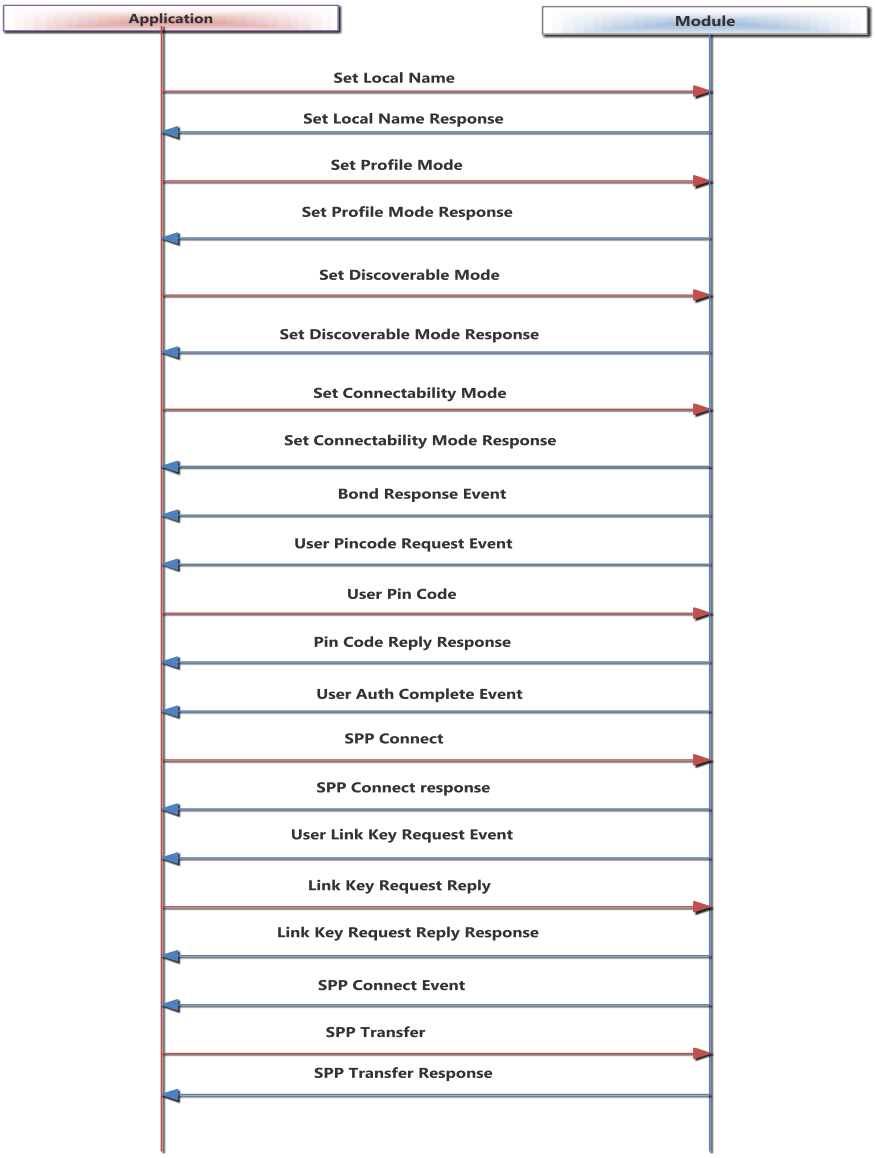

Figure 7 Sample flow in BT Slave Mode and do SPP Tx

6.5 AT command sequence to perform SPP data transfer in BT Master mode

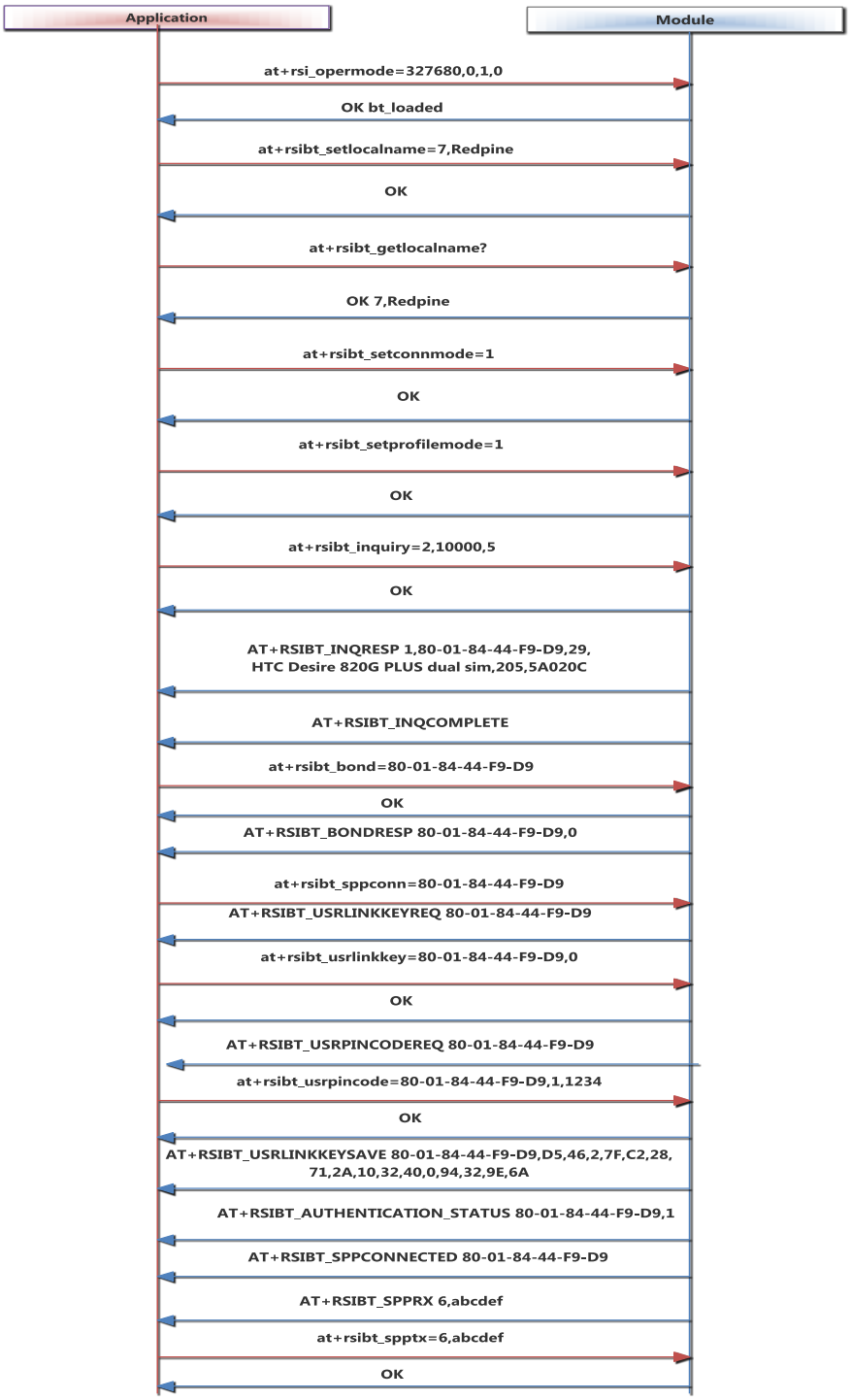

Figure 8 AT command flow in BT Master mode

6.6 AT command sequence to perform SPP data transfer in BT Slave mode

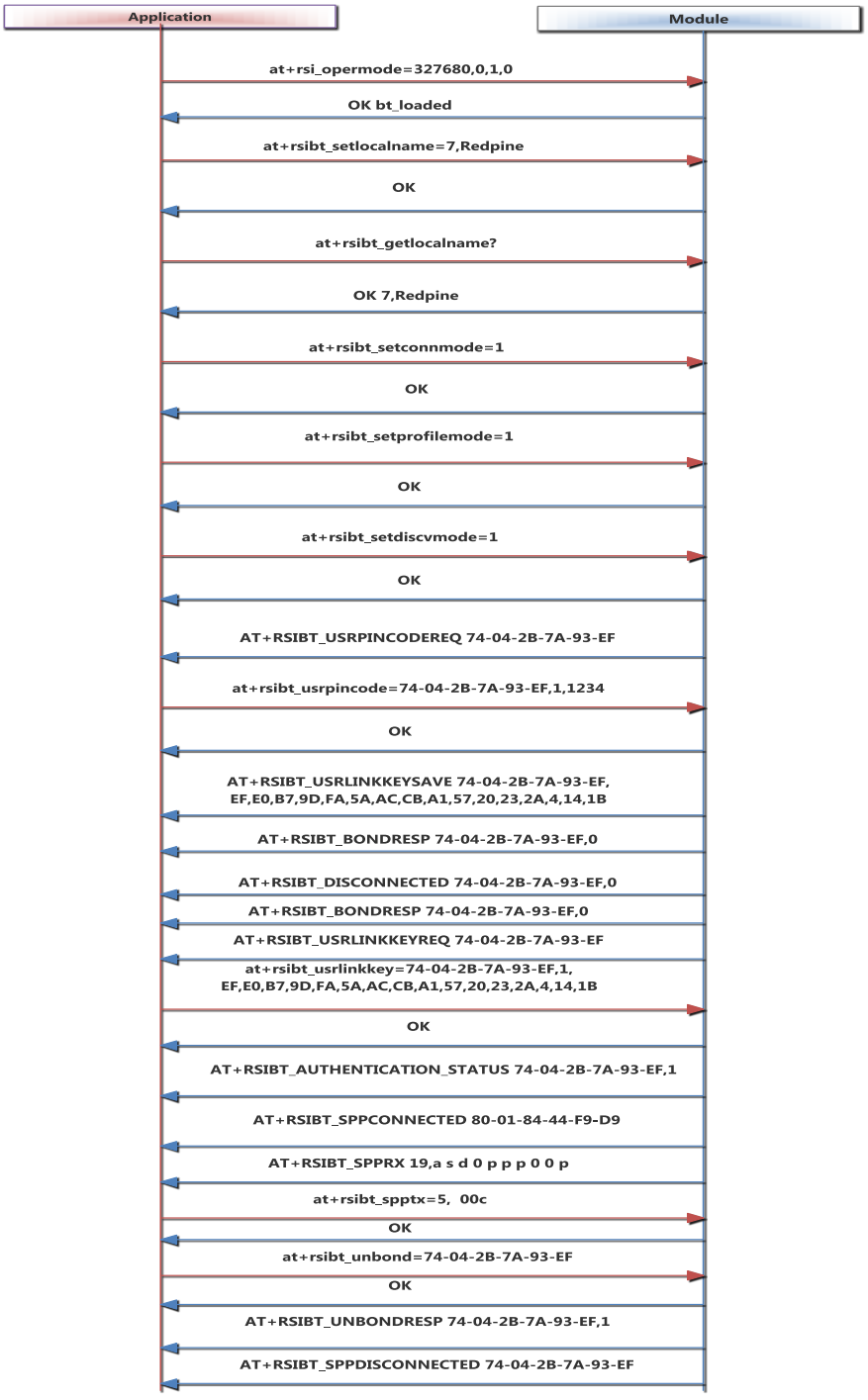

Figure 9 AT command flow in BT Slave mode

Revision History

Revision No.	Version No.	Date	Changes
1	1.0	Oct 2014	Added BT classic Section
2	1.1	Nov 2014	Added Init, Deinit, Antenna Select & Config Enable commands to BT classic Section.
3	1.0.10lc	Feb 2014	1.SPP Transfer and Receive payload size is restricted to 200 bytes. 2.Updated pincode reply,SPP_TX,SPP_RX event structures. 3.Added Linkkey reply command&Linkkey save,Linkkey request events 4.Updated the flow diagrams.
3	1.1.0	Mar 2015	Deleted config enable command.
4	1.1.1	Apr 2015	Deleted operations through host interface section
5	1.3.0	July 2015	Added AT commands format and IAP commands.
6	1.3.1	Aug 2015	Added bt init,bt deinit, and Antenna select functions AT command format. Missed function prototypes.
7	1.4.0	Nov 2015	Added enable SSP mode command.
8	1.4.1	Dec 2015	Added sniff commands and events. Added AT commands sequence in Appendix. Format changes of IAP commands and events
9	1.5.0	March 2016	Added 0X4E39 Error Code Wrong BD Address. Added NOTE in SPP commands.
10	1.5.0	April 2016	Added PER Commands. Corrected Command ID's.