RUI API Reference

Version V1.0 | August 22, 2019

www.RAKwireless.com

Visit our website for more document.

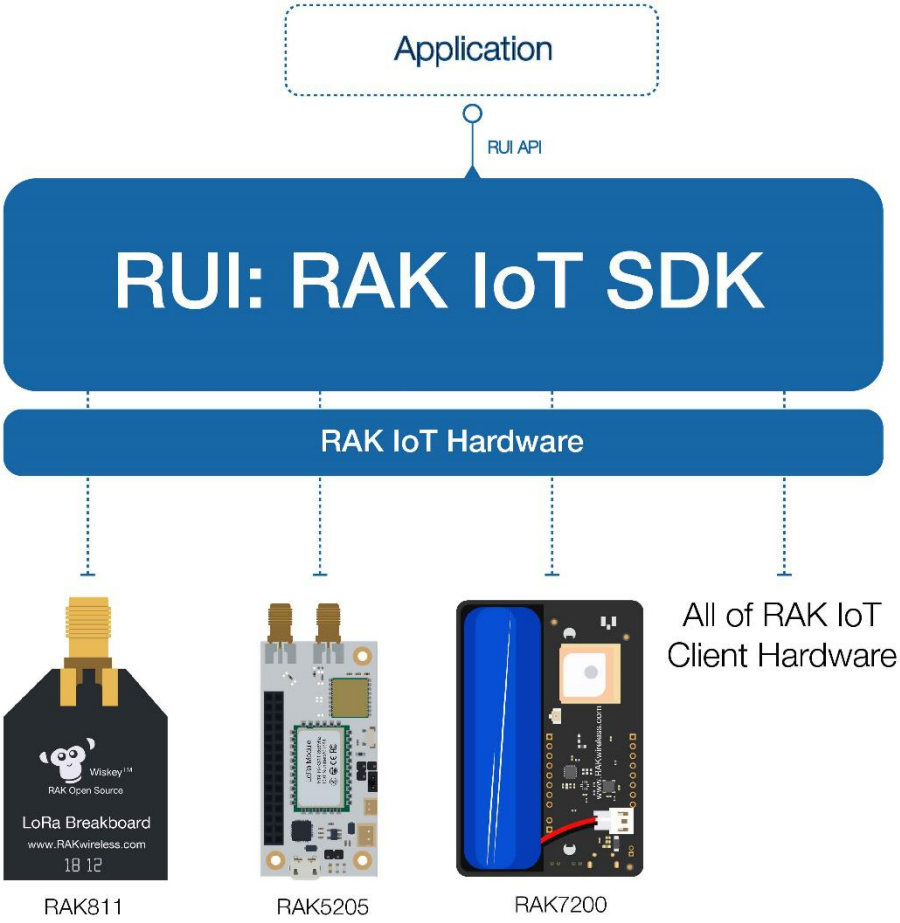
Table of Contents

1. Overview....4
2. Cellular....5
1.1 rui_cellular_send....5
1.2 rui_cellular_response....5
1.3 rui_cellular_register_recv_callback....5
1.4 rui_cellular_set_mode....6
1.5 rui_cellular_open_socket....6
1.6 rui_cellular_set_operator....6
3. LoRa....7
2.1 rui_lora_send....8
2.2 rui_lora_register_recv_callback....8
2.3 rui_lora_set_device_mode....9
2.4 rui_lora_set_dev_eui....9
2.5 rui_lora_set_app_eui....9
2.6 rui_lora_set_app_key....10
2.7 rui_lora_set_dev_addr....10
2.8 rui_lora_set_apps_key....10
2.9 rui_lora_set_nwks_key....11
2.10 rui_lora_set_channel_mask....11
2.11 rui_lora_class....11
2.12 rui_lora_set_dr....11
2.13 rui_lora_set_join_mode....12
2.14 rui_lora_set_work_mode....12
2.15 rui_lora_set_send_interval....12
2.16 rui_lora_set_region....13
2.17 rui_lora_get_status....13
2.18 rui_lorajoin_register_callback....13
2.19 rui_lorasend_complete_register_callback....13
2.20 rui_lora_tx_power....14
2.21 rui_lorap2p_config....14
2.22 rui_lorap2p_send....15
2.23 rui_lorap2p_register_recv_callback....15
4. BLE....15
3.1 rui_ble_scan_adv....16
3.2 rui_ble_rx_data_notify....17
3.3 rui_ble_rx_data_read....17
3.4 rui_ble_tx_data_write....17
3.5 rui_ble_tx_data_read....18
5. Interface....18
4.1 rui_uart_send....18
4.2 rui_uart_recv....19
4.3 rui_gpio_rw....19

4.4 rui_adc_read....19
4.5 rui_iic_rw....20
6. Device....20
5.1 rui_device_version....20
5.2 rui_device_reset....21
5.3 rui_device_sleep....21
5.4 rui_device_boot....21
7. Timer....22
6.1 rui_timer_init....22
6.2 rui_timer_setvalue....22
6.2 rui_timer_start....22
6.3 rui_timer_stop....23
6.4 rui_delay_ms....23

1. Overview

This document is the API reference of RUI.
You can develop an application as you want very easily and elegantly on RAK IoT products and modules by using these APIs.



Global definition:

```
#define SUCCESS          1
#define FAIL             0
typedef enum DRIVER_MODE
{
    RUI_NORMAL_MODE = 0,
    RUI_POWER_ON_MODE,
    RUI_POWER_OFF_MODE,
    RUI_SLEEP_MODE,
    RUI_STANDBY_MODE
}DRIVER_MODE;
```

2. Cellular

General format: **rui_cellular_xxx()**

1.1 rui_cellular_send

uint32_t rui_cellular_send(uint8_t *data)

```
*********************************************************************************
 *   @brief      This API is used to send data through cellular.
 *   @return     SUCCESS or FAIL
 *   @param      uint8_t *data:      the data which will be sent through cellular.
 *   @support    RAK8212-M and RAK5010 core module.
 *********************************************************************************
```

1.2 rui_cellular_response

uint32_t rui_cellular_response(uint8_t *response, uint32_t len, uint32_t timeout)

```
*********************************************************************************
 *   @brief      This API is used to wait for a correct response from cellular module.
 *   @return     SUCCESS or FAIL
 *   @param      uint8_t *response:      the response of server string
                 uint32_t len:           response data length
                 uint32_t timeout:       the time to wait for response until timeout.
 *   @support    RAK8212-M and RAK5010 core module.
*********************************************************************************
```

1.3 rui_cellular_register_recv_callback

typdef void (*cellular_receive)(uint8_t *data);
uint32_t rui_cellular_register_recv_callback(cellular_receive callback)

```
*********************************************************************************
 *    @brief      This API is used to register a callback function for cellular in application
                  so that application can receive cellular data automatically.
 *    @return     SUCCESS or FAIL
 *    @param      cellular_receive callback:  the callback function for receiving cellular
data.
 *   @support    RAK8212-M and RAK5010 core module.
*********************************************************************************
```

1.4 rui_cellular_set_mode

```
uint32_t rui_cellular_set_mode(DRIVER_MODE mode)
*********************************************************************************
 *   @brief      This API is used to set the work mode of the cellular module.
 *   @return     SUCCESS or FAIL
 *   @param      DRIVER_MODE mode:   the cellular module's work mode
 *   @support    RAK8212-M and RAK5010 core module.
 *********************************************************************************
```

1.5 rui_cellular_open_socket

```
uint32_t rui_cellular_open_socket(uint8_t* data)
*********************************************************************************
*    @brief      This API is used to open a TCP socket with a remote server.
*    @return     SUCCESS or FAIL
*    @param      uint8_t* data:      open TCP link cmd, for example:
                                     AT+QIOPEN=1,0,"TCP","ip",%port,0,1"
*    @support    RAK8212-M and RAK5010 core module.
*********************************************************************************
```

1.6 rui_cellular_set_operator

```
uint32_t rui_cellular_set_operator(uint8_t *APN,uint8_t *operator_long_name,
                                   uint8_t *operator_short_name,uint8_t operator_net)
*********************************************************************************
*    @brief      This API is used to configure parameters of Cellular operator.
*    @return     SUCCESS or FAIL
*    @param      uint8_t *APN:                   The APN name of Cellular operator.
                 uint8_t *operator_long_name:        The operator's long name.
                                                 For example, "CHINA MOBILE".
                 uint8_t *operator_short_name:   The operator's short name.
                                                 For example, "CMCC".
                 uint8_t operator_net:               The cellular network type.
*    @support    RAK8212-M and RAK5010 core module.
*********************************************************************************
```

3. LoRa

General format: **rui_lora_xxx()**
General definition:

```
typedef enum LORA_JOIN_MODE
{
    RUI_OTAA = 0,
    RUI_ABP
}LORA_JOIN_MODE;

typedef enum LORA_CLASS_MODE
{
    CLASS_A = 0,
    CLASS_B,
    CLASS_C
}LORA_CLASS_MODE;

typedef enum LORA_WORK_MODE
{
    RUI_LORAWAN = 0,
    RUI_P2P,
    RUI_TESTMODE
}LORA_WORK_MODE;

typedef enum LORA_REGION
{
    AS923,
    AU915,
    CN470,
    CN779,
    EU433,
    EU868,
    IN865,
    KR920,
    US915,
    US915_Hybrid
}LORA_REGION;

typedef struct RUI_RECEIVE
{
uint8_t Port;           //Application port
uint8_t RxDatarate;         // Downlink datarate
uint8_t *Buffer;        //Pointer to the received data stream
```

```
uint8_t BufferSize;          // Size of the received data stream
int16_t Rssi;               //RSSI of the received packet
uint8_t Snr;                // SNR of the received packet
uint32_t DownLinkCounter;      // The downlink counter value for the received frame
}RUI_RECEIVE_T;

typedef struct RUI_LORAP2P_RECEIVE
{
uint8_t *Buffer;           // Pointer to the received data stream
uint8_t BufferSize;          // Size of the received data stream
int16_t Rssi;               // Rssi of the received packet
uint8_t Snr;                // Snr of the received packet
}RUI_LORAP2P_RECEIVE_T;

typedef enum RUI_EMCPS
{
RUI_MCPS_UNCONFIRMED,        // Unconfirmed LoRaMAC frame
RUI_MCPS_CONFIRMED,          // Confirmed LoRaMAC frame
RUI_MCPS_MULTICAST,        // Multicast LoRaMAC frame
RUI_MCPS_PROPRIETARY,        // Proprietary frame
RUI_LORAEVENT_ERROR          // LoRa event error
}RUI_MCPS_T;
```

2.1 rui_lora_send

```
uint32_t rui_lora_send(uint8_t port,uint8_t* data,uint8_t len)

*********************************************************************************************
 *   @brief        rui_lora_send
 *   @return       SUCCESS or FAIL
 *   @param        uint8_t port:   send data port
                   uint8_t* data: send data string
                   uint8_t len:    send data length
 *   @support      RAK811, RAK4200, and RAK4600 core module.
*********************************************************************************************
```

2.2 rui_lora_register_recv_callback

```
typdef void (*lora_receive)( RUI_RECEIVE_T *data);
uint32_t rui_lora_register_recv_callback(lora_receive callback)
```

```
*******************************************************************************************
 *    @brief        This API is used to register a callback function for LoRa in application,
                    so that application can receive the LoRa data automatically.
 *    @return       SUCCESS or FAIL
 *    @param        lora_receive callback:      the callback function for receiving LoRa
data.
                    RUI_RECEIVE_T *data:    the received data.
 *    @support      RAK811, RAK4200, and RAK4600 core module.
*******************************************************************************************
```

2.3 rui_lora_set_device_mode

uint32_t rui_lora_set_device_mode(DRIVER_MODE mode)

```
*******************************************************************************************
 *    @brief        This API is used to set the work mode of LoRa module.
 *    @return       SUCCESS or FAIL
 *    @param        DRIVER_MODE mode:    lora peripheral work mode
 *    @support      RAK811, RAK4200, and RAK4600 core module.
*******************************************************************************************
```

2.4 rui_lora_set_dev_eui

uint32_t rui_lora_set_dev_eui(uint8_t *dev_eui)

```
*******************************************************************************************
 *    @brief        This API is used to set the device EUI for LoRaWAN OTAA mode.
 *    @return       SUCCESS or FAIL
 *    @param        uint8_t *dev_eui:  the device EUI.
 *    @support      RAK811, RAK4200, and RAK4600 core module.
*******************************************************************************************
```

2.5 rui_lora_set_app_eui

uint32_t rui_lora_set_app_eui(uint8_t *app_eui)

```
*******************************************************************************************
 *    @brief        This API is used to set the application EUI for LoRaWAN OTAA mode.
```

```
 *    @return        SUCCESS or FAIL
 *    @param         uint8_t *app_eui:        the application EUI.
 *    @support       RAK811, RAK4200, and RAK4600 core module.
*******************************************************************************************
```

2.6 rui_lora_set_app_key

uint32_t rui_lora_set_app_key(uint8_t *app_key)

```
*******************************************************************************************
 *    @brief         This API is used to set the application key for LoRaWAN OTAA mode.
 *    @return        SUCCESS or FAIL
 *    @param         uint8_t *app_key:        the application key.
 *    @support       RAK811, RAK4200, and RAK4600 core module.
*******************************************************************************************
```

2.7 rui_lora_set_dev_addr

uint32_t rui_lora_set_dev_addr(uint8_t *dev_addr)

```
*******************************************************************************************
 *    @brief         This API is used to set the device address for LoRaWAN ABP mode.
 *    @return        SUCCESS or FAIL
 *    @param         uint8_t *dev_addr:       the device address.
 *    @support       RAK811, RAK4200, and RAK4600 core module.
*******************************************************************************************
```

2.8 rui_lora_set_apps_key

uint32_t rui_lora_set_apps_key(uint8_t *apps_key)

```
*******************************************************************************************
 *    @brief         This API is used to set the application session key for LoRaWAN ABP
mode.
 *    @return        SUCCESS or FAIL
 *    @param         uint8_t *apps_key: the application session key.
 *    @support       RAK811, RAK4200, and RAK4600 core module.
*******************************************************************************************
```

2.9 rui_lora_set_nwks_key

uint32_t rui_lora_set_nwks_key(uint8_t *nwks_key)

```
*******************************************************************************************
 *   @brief        This API is used to set the network session key for LoRaWAN ABP mode.
 *   @return       SUCCESS or FAIL
 *   @param        uint8_t *nwks_key:      the network session key.
 *   @support      RAK811, RAK4200, and RAK4600 core module.
*******************************************************************************************
```

2.10 rui_lora_set_channel_mask

uint32_t rui_lora_set_channel_mask(uint8_t channel, uint8_t on_off)

```
*******************************************************************************************
 *   @brief        This API is used to turn a certain channel on or off.
 *   @return       SUCCESS or FAIL
 *   @param        uint8_t channel:    the channel number you want to set.
                   Uint8_t on_off:     turn on or turn off.
 *   @support      RAK811, RAK4200, and RAK4600 core module.
*******************************************************************************************
```

2.11 rui_lora_class

uint32_t rui_lora_set_class(LORA_CLASS_MODE class)

```
*******************************************************************************************
 *   @brief        This API is used to set the LoRaWAN Class.
 *   @return       SUCCESS or FAIL
 *   @param        LORA_CLASS_MODE class: Class A, Class B, or Class C.
 *   @support      RAK811, RAK4200, and RAK4600 core module.
*******************************************************************************************
```

2.12 rui_lora_set_dr

uint32_t rui_lora_set_dr(uint8_t dr);

```
*******************************************************************************************
 *   @brief        This API is used to set the DR for LoRa node.
 *   @return       SUCCESS or FAIL
 *   @param        uint8_t dr:     the value of DR.
 *   @support      RAK811, RAK4200, and RAK4600 core module.
*******************************************************************************************
```

2.13 rui_lora_set_join_mode

uint32_t rui_lora_set_join_mode(LORA_JOIN_MODE mode)

```
*******************************************************************************************
 *   @brief        This API is used to set the join mode of LoRaWAN.
 *   @return       SUCCESS or FAIL
 *   @param        LORA_JOIN_MODE mode:      OTAA or ABP
 *   @support      RAK811, RAK4200, and RAK4600 core module.
*******************************************************************************************
```

2.14 rui_lora_set_work_mode

uint32_t rui_lora_set_work_mode(LORA_WORK_MODE mode)

```
*******************************************************************************************
 *   @brief        This API is used to set the work mode of LoRa module.
 *   @return       SUCCESS or FAIL
 *   @param        LORA_WORK_MODE mode:    LaRaWAN, P2P, or Test mode.
 *   @support      RAK811, RAK4200, and RAK4600 core module.
*******************************************************************************************
```

2.15 rui_lora_set_send_interval

uint32_t rui_lora_set_send_interval(uint16_t interval_time)

```
*******************************************************************************************
 *   @brief        This API is used to set the interval time of sending data.
 *   @return       SUCCESS or FAIL
 *   @param        uint16_t app_interval:        the interval time of sending data.
 *   @support      RAK811, RAK4200, and RAK4600 core module.
*******************************************************************************************
```

2.16 rui_lora_set_region

uint32_t rui_lora_set_region(LORA_REGION region)

```
*********************************************************************************************
 *   @brief        This API is used to set the region of LoRaWAN you want it to work in.
 *   @return       SUCCESS or FAIL
 *   @param        LORA_REGION region:     the region of LoRaWAN.
 *   @support      RAK811, RAK4200, and RAK4600 core module.
*********************************************************************************************
```

2.17 rui_lora_get_status

uint32_t rui_lora_get_status(uint8_t *status)

```
*********************************************************************************************
 *   @brief        This API is used to get all status about LoRa.
 *   @return       SUCCESS or FAIL
 *   @param        uint8_t *status:     the status about LoRa.
 *   @support      RAK811, RAK4200, and RAK4600 core module.
*********************************************************************************************
```

2.18 rui_lorajoin_register_callback

typedef void (*lorajoin)(uint32_t status);
uint32_t rui_lorajoin_register_callback(lorajoin callback)

```
*********************************************************************************************
 *   @brief        This API is used to register a callback function for LoRaWAN join result.
 *   @return       SUCCESS or FAIL
 *   @param        lorajoin callback:      the callback function for LoRaWAN join
successed.
                   status:                 1 -> join succeed, 0 -> join fail.
 *   @support      RAK811, RAK4200, and RAK4600 core module.
*********************************************************************************************
```

2.19 rui_lorasend_complete_register_callback

typedef void (*lorasend)(RUI_MCPS_T type);

uint32_t rui_lorasend_complete_register_callback(lorasend callback)
****
* @brief This API is used to register a callback function for LoRaWAN send complete,
so that application can send the next packet.
* @return SUCCESS or FAIL
* @param lorasend callback: the callback function.
RUI_MCPS_T type: the packet type.
* @support RAK811, RAK4200, and RAK4600 core module.
****

2.20 rui_lora_tx_power

uint32_t rui_lora_tx_power(uint8_t power_value)
****
* @brief This API is used to set lora TX power.
* @return SUCCESS or FAIL
* @param uint8_t power_value: the TX power level, more details please have a look
at LoRaWAN 1.0.2 region specification.
https://github.com/RAKWireless/Update-File/blob/master/LoRaWANRegionalParametersv1.0.2.pdf
* @support RAK811, RAK4200, and RAK4600 core module.
****

2.21 rui_lorap2p_config

uint32_t rui_lorap2p_config(uint32_t Frequency,uint8_t Spreadfact,uint8_t Bandwidth,uint8_t Codingrate,uint16_t Preamlen,uint8_t Powerdbm)
****
* @brief This API is used to config LoRaP2P parameters.
* @return SUCCESS or FAIL
* @param Frequency: Frequency in Hz
Spreadfact: Spreadfactare limite to the 6-12 range
Bandwidth: Bandwidth limite to the 0-2 range
Codingrate: Codingrate limite to the 1-4 range
Preamlen: Preamlen limite to the 2-66535 range
Powerdbm: Powerdbm limite to the 0-20 range.
* @support RAK811, RAK4200, and RAK4600 core module.
****

2.22 rui_lorap2p_send

uint32_t rui_lorap2p_send(uint8_t* data,uint16_t len)

```
*********************************************************************************************
 *     @brief        This API is used to send data by LoRaP2P mode.
 *     @return       SUCCESS or FAIL
 *     @param            data :     the data you want to send.
                       len:        the data length.
 *   @support       RAK811, RAK4200, and RAK4600 core module.
*********************************************************************************************
```

2.23 rui_lorap2p_register_recv_callback

typedef void (*lorap2p_receive)(RUI_LORAP2P_RECEIVE_T *data);
uint32_t rui_lorap2p_register_recv_callback(lorap2p_receive callback)

```
*********************************************************************************************
  *    @brief          This API is used to register a callback function for LoRaP2P, so that application
                    can receive the LoRap2p data automatically.
 *    @return       SUCCESS or FAIL
 *    @param        lora_receive callback:    the callback function for receiving LoRaP2P data.
 *   @support       RAK811, RAK4200, and RAK4600 core module.
*********************************************************************************************
```

4. BLE

General format: **rui_ble_xxx()**
General definition:

```
typedef enum
{
      BLE_CLIENT_EVT_DISCOVERY_COMPLETE = 1,
      BLE_CLIENT_EVT_NOTIFICATION,
      BLE_CLIENT_EVT_READ
} BLE_CLIENT_EVT_TYPE;

typedef struct
{
   uint8_t        *p_data;
```

```
    uint16_t        len;
} BLE_DATA;

typedef struct
{
    uint16_t write_handle;
    uint16_t read_notify_handle;
    uint16_t notify_cccd_handle;
} BLE_HANDLE;

typedef struct
{
      BLE_CLIENT_EVT_TYPE    evt_type;
      uint16_t    conn_handle;
      union
      {
            BLE_DATA          data;
            BLE_HANDLE      peer_db;
      } params;
} BLE_CLIENT_EVT;

typedef void (* BLE_CLIENT_EVT_HANDLER) (BLE_CLIENT * p_ble_rcs_c,
BLE_CLIENT_EVT * p_evt);

/**@brief Rak Custom Service Client structure. */
struct BLE_CLIENT
{
      uint16_t     connect_handle;
      HANDLE_LIST     peer_list;
      BLE_CLIENT_EVT_HANDLER evt_handler;
      uint8_t       uuid_type;
};
```

3.1 rui_ble_scan_adv

```
void rui_ble_scan_adv(int8_t rssi_value, uint8_t *p_adv_data, uint16_t adv_data_len,
uint8_t *p_device_mac)
*********************************************************************************
 *   @brief         This API is used to scan all around BLE devices.
                    This API is valid when the BLE works on master mode.
 *   @return        NULL
 *   @param         int8_t rssi_value:              peripheral rssi value.
 *   @param         uint8_t *p_adv_data:                the advertise data.
```

```
 *    @param        uint16_t adv_data_len:     the length of advertise data.
 *    @param        uint8_t *p_device_mac:     the peripheral's MAC address.
 *    @support      RAK8212-M, RAK5010, and RAK4600 core module.
*********************************************************************************************
```

3.2 rui_ble_rx_data_notify

```
void rui_ble_rx_data_notify(uint8_t *pdata, uint16_t len)
*********************************************************************************************
 *    @brief        This API is used to receive the notify data which come from other devices
                    through BLE.
 *    @return       NULL
 *    @param        uint8_t *pdata:    the receive data.
 *    @param        uint16_t len:      the length of receive data.
 *    @support      RAK8212-M, RAK5010, and RAK4600 core module.
*********************************************************************************************
```

3.3 rui_ble_rx_data_read

```
void rui_ble_rx_data_read(uint8_t *pdata, uint16_t len)
*********************************************************************************************
 *    @brief        This API is used to receive the data which you just read from another BLE
                    device through BLE.
 *    @return       NULL
 *    @param        uint8_t *pdata:    the receive data.
 *    @param        uint16_t len:            the length of receive data.
 *    @support      RAK8212-M, RAK5010, and RAK4600 core module.
*********************************************************************************************
```

3.4 rui_ble_tx_data_write

```
uint32_t rui_ble_tx_data_write(BLE_CLIENT * p_ble_rcs_c, uint8_t *pdata, uint16_t len)
*********************************************************************************************
 *    @brief        This API is used to write a data to another BLE device through BLE.
 *    @return       SUCCESS or FAIL
 *    @param        BLE_CLIENT * p_ble_rcs_c:     The BLE client instance.
 *    @param        uint8_t *pdata:               The data which will be sent.
```

```
 *   @param        uint16_t len:                    The length of data.
 *   @support     RAK8212-M, RAK5010, and RAK4600 core module.
*******************************************************************************************
```

3.5 rui_ble_tx_data_read

```
uint32_t rui_ble_tx_data_read(BLE_CLIENT * p_ble_rcs_c)
*******************************************************************************************
 *   @brief        This API is used to read another BLE device's data through BLE.
 *   @return       SUCCESS or FAIL.
 *   @param        BLE_CLIENT * p_ble_rcs_c:   The BLE client instance.
 *   @support     RAK8212-M, RAK5010, and RAK4600 core module.
*******************************************************************************************
```

5. Interface

General format: **rui_<Interface type>_xxx()**
General definition:

```
typedef enum RUI_UART_DEF
{
    RUI_UART0 = 0,
    RUI_UART1 = 1,
    RUI_UART2 = 2,
    RUI_UART3 = 3
}RUI_UART_DEF;

Typedef enum RUI_IF_READ_WRITE
{
    RUI_IF_READ = 0,
    RUI_IF_WRITE = 1
}RUI_IF_READ_WRITE;
```

4.1 rui_uart_send

```
uint32_t rui_uart_send(RUI_UART_DEF uart_def, uint8_t *pdata, uint16_t len)
*******************************************************************************************
 *   @brief            This API is used to send data via uart.
 *   @return           SUCCESS or FAIL
```

```
 *   @param          RUI_UART_DEF uart_def:      the instance of uart.
 *   @param          uint8_t *pdata:             the pointer of data you want to send.
 *   @param          uint16_t len:               the length of data.
 *   @support        RAK811, RAK4200, RAK8212-M, RAK5010, and RAK4600 core
module.
*******************************************************************************
```

4.2 rui_uart_recv

```
void rui_uart_recv(RUI_UART_DEF uart_def, uint8_t *pdata, uint16_t len)
*******************************************************************************
 *   @brief          This API is used to receive data from uart.
 *   @return         NULL
 *   @param          RUI_UART_DEF uart_def:      the instance of uart.
 *   @param          uint8_t *pdata:             the pointer of data.
 *   @param          uint16_t len:               the length of data. This value is always
1.
 *   @support        RAK811, RAK4200, RAK8212-M, RAK5010, and RAK4600 core
module.
*******************************************************************************
```

4.3 rui_gpio_rw

```
uint32_t rui_gpio_rw(RUI_IF_READ_WRITE rw, uint8_t pin, uint8_t* status)
*******************************************************************************
 *   @brief      This API is used to read or set a certain gpio's status.
 *   @return     SUCCESS or FAIL
 *   @param      RUI_IF_READ_WRITE rw: read or set.
                 uint8_t pin: the pin number of gpio which you want to read or set.
                 uint8_t* status:    the pointer of gpio value.
                                     0: low level, 1: high level.
 *   @support        RAK811, RAK4200, RAK8212-M, RAK5010, and RAK4600 core
module.
*******************************************************************************
```

4.4 rui_adc_read

```
uint32_t rui_adc_read(uint8_t pin, uint8_t* value)
*******************************************************************************
```

```
 *   @brief        This API is used to read the value of ADC pin
 *   @return       SUCCESS or FAIL
 *   @param        uint8_t pin:        the pin number of ADC.
                   uint8_t* value:     the value which is read from ADC.
 *   @support          RAK811, RAK4200, RAK5010, and RAK4600 core module.
*******************************************************************************
```

4.5 rui_iic_rw

uint32_t rui_iic_rw(RUI_IF_READ_WRITE rw, uint8_t devAddr, uint16_t regAddr, uint8_t* data, uint16_t len)

```
*******************************************************************************
 *   @brief        This API is used to read/write through I2C.
 *   @return       SUCCESS or FAIL
 *   @param        RUI_IF_READ_WRITE rw:       read or write through I2C.
                   uint8_t devAddr:            device address, in Hex format.
                   uint16_t regAddr:           the sensor's register address, in Hex format.
                   uint8_t* data:              the data read out or will be write through iic.
                   uint16_t len:               the data length. the unit is byte.
 *   @support          RAK811, RAK4200, RAK8212-M, RAK5010, and RAK4600 core module.
*******************************************************************************
```

6. Device

General format: **rui_device_xxx()**

5.1 rui_device_version

uint32_t rui_device_version(uint8_t *version)

```
*******************************************************************************
 *   @brief        This API is used to get the current firmware version.
 *   @return       SUCCESS or FAIL
 *   @param        uint8_t *version:   the current firmware version.
 *   @support          RAK811, RAK4200, RAK8212-M, RAK5010, and RAK4600 core module.
*******************************************************************************
```

5.2 rui_device_reset

uint32_t rui_device_reset(void)

```
*******************************************************************************************
 *   @brief        This API is used to reset the device.
 *   @return       SUCCESS or FAIL
 *   @param        void
 *   @support          RAK811, RAK4200, RAK8212-M, RAK5010, and RAK4600 core
module.
*******************************************************************************************
```

5.3 rui_device_sleep

void rui_device_sleep(uint32_t period)

```
*******************************************************************************************
 *   @brief        This API is used to let the device go to sleep mode.
 *   @return       NULL
 *   @param        uint32_t period:    sleep period
 *   @support          RAK811, RAK4200, RAK8212-M, RAK5010, and RAK4600 core
module.
*******************************************************************************************
```

5.4 rui_device_boot

void rui_device_boot()

```
*******************************************************************************************
 *   @brief        This API is used to let the device go to boot mode.
 *   @return       NULL
 *   @param        NULL
 *   @support          RAK811, RAK4200, RAK8212-M, RAK5010, and RAK4600 core
module.
*******************************************************************************************
```

7. Timer

General format: **rui_timer_xxx()**

6.1 rui_timer_init

uint32_t rui_timer_init(void *obj, void (*callback)(void))
****
 * @brief This API is used to create/initialize a timer.
 * @return SUCCESS or FAIL
 * @param void *obj: timer instance.
 void (*callback)(void): timer event callback function.
 * @support RAK811, RAK4200, RAK8212-M, RAK5010, and RAK4600 core module.
****

6.2 rui_timer_setvalue

uint32_t rui_timer_setvalue(void *obj, uint32_t value)
****
 * @brief This API is used to set the interval value for timer.
 * @return SUCCESS or FAIL
 * @param void *obj: timer instance.
 uint32_t value： timer value.
 * @support RAK811, RAK4200, RAK8212-M, RAK5010, and RAK4600 core module.
****

6.2 rui_timer_start

uint32_t rui_timer_start(void *obj)
****
 * @brief This API is used to start the timer.
 * @return SUCCESS or FAIL
 * @param void *obj: timer instance.
 * @support RAK811, RAK4200, RAK8212-M, RAK5010, and RAK4600 core module.

****

6.3 rui_timer_stop

uint32_t rui_timer_stop(void *obj)

****

```
 *     @brief        This API is used to stop the timer.
 *     @return       SUCCESS or FAIL
 *     @param            void *obj:      timer instance.
 *   @support     RAK811, RAK4200, RAK8212-M, RAK5010, and RAK4600 core module.
```

****

6.4 rui_delay_ms

uint32_t rui_delay_ms(uint32_t value)

****

```
 *     @brief        This API is used to delay some ms.
 *     @return       SUCCESS or FAIL
 *     @param            uint32_t value:     delay some ms.
 *   @support     RAK811, RAK4200, RAK8212-M, RAK5010, and RAK4600 core module.
```

****