Identium F529 Reader Library (RFID_READER_DLL) - API Guide

Identium F529 Active Reader SDK 1.0 | Version 2.0 of the reader library | (c) 2026 Identium Tech Solutions Pvt. Ltd. | info@identium.in

Note: dll is developed in C language, extern "C" is compiled and linked according to C language

```
/* Error code definition: */
#define PARAMETER_ERROR     (-101)      // Function parameter error.
#define INVALID_BUFFER      (-102)      // Receive invalid buffer because CRC16 check error.
#define JSON_PARSING_ERROR  (-103)      // Json string parameter parsing failure.
```

```
/**
   Read the command code initiated by the device and call the corresponding function according to the command code

   @param recv_buffer The network receives the data packet reported by the device
   @param recv_len Specify the number of bytes of recv_buffer
    @param json_result(Export parameter) The message header information result, the format is JSON string, for example:
   {
     "cmd" : "0x0001",              // command code
     "seqno" : "00000000",          // Message serial number
     "device" : "861694034205896",  // device ID
     "version": "3.6"               // Device software version number, empty if not
   }
   @param result_length(Out parameter) specify the length of json_result result
    @return Return the command code requested by the device; return a negative number on error (see error code definition)
```

```
*/
RFID_DLL_API int RFID_ReadCmd(const unsigned char *recv_buffer, const int recv_len,
                              char *json_result, int *result_length);


/**
  The device initiates registration with the platform cmd: 0x0008

  @param recv_buffer The network receives the data packet reported by the device
  @param recv_len Specify the number of bytes of recv_buffer
  @param respond_cmd The platform responds to the command: 0x00--registration successful, 0xFF--registration rejected
  @param json_str_server_addr The platform responds to the IP and port of the load server (valid only after successful registration), the format is a JSON string, for example:
  {
    "ip" : "192.168.1.100",       // Load server IP
    "port" : 4600                 // Load server port
  }
  @param send_buffer(Export reference) The data packet to be sent by the platform to the device (the length is limited to 250 bytes)
  @return Returns the size of the data packet to be sent to the device; returns a negative number on error (see the definition of error code)
*/
RFID_DLL_API int RFID_Register(const unsigned char *recv_buffer, const int recv_len,
                               const unsigned char respond_cmd,
                               const char *json_str_server_addr,
                               unsigned char *send_buffer);


/**
  The device initiates a login to the platform cmd: 0x0001

  @param recv_buffer The network receives the data packet reported by the device
  @param recv_len Specify the number of bytes of recv_buffer
  @param respond_cmd Platform response command: 0x00--login successful, 0xFF--login refused
  @param send_buffer(Export reference) The data packet to be sent by the platform to the device (the length is limited to 250 bytes)

  @return Returns the size of the data packet to be sent to the device; returns a negative number on error (see the definition of error code)*/
RFID_DLL_API int RFID_Login(const unsigned char *recv_buffer, const int recv_len,
                            const unsigned char respond_cmd,
                            unsigned char *send_buffer);
```

```
/**
  The device initiates a heartbeat to the platform cmd: 0x0003

  @param recv_buffer The network receives the data packet reported by the device
  @param recv_len Specify the number of bytes of recv_buffer
  @param respond_cmd Platform response command: 0x00--no operation instruction
  @param send_buffer (Export reference) The data packet to be sent by the platform to the device (the length is limited to 250 bytes)

  @return Returns the size of the data packet to be sent to the device; returns a negative number on error (see the definition of error code)
*/
RFID_DLL_API int RFID_Heartbeat(const unsigned char *recv_buffer, const int recv_len,
                                const unsigned char respond_cmd,
                                unsigned char *send_buffer);


/**
  The device initiates a data report to the platform cmd: 0x0004

  @param recv_buffer The network receives the data packet reported by the device
  @param recv_len Specify the number of bytes of recv_buffer
  @param send_buffer(Export reference) The data packet to be sent by the platform to the device (the length is limited to 250 bytes)

  @return Returns the size of the data packet to be sent to the device; returns a negative number on error (see the definition of error code)
*/
RFID_DLL_API int RFID_DataRecord(const unsigned char *recv_buffer, const int recv_len,
                                 unsigned char *send_buffer);


/**
  Analyze the received tag data cmd: 0x0004

  @param recv_buffer The network receives the data packet reported by the device
  @param recv_len Specify the number of bytes of recv_buffer
  @param json_result (Output parameter) The tag data result obtained after parsing, the format is a JSON array, for example:
  {
      "TAG":[
      {
     "device": "861694034205706",     //device ID
```

```
        "tagid": "16100001",              //tag ID
        "tlvtype": "0x8B01",              //TLV type
        "tagtype": "20",                  //tag type
           "tagdata": "36.2",                 //The tag  carries the data value (current unit: mA, temperature unit: ℃)
        "antenna": 1,                     //Antenna number, 1,2,3,4 correspond to 4 antennas in east, south, west and north respectively
        "intensity": -78,                 //RSSI, -78 dBm
            "entry": 0,                       //Status of entering and exiting the base station: 1: Entering the base station reading range, 0: Out of the base station reading range
            "staying": 0,                     //Base station staying flag: 1: base station staying, 0: not base station staying
        "alarm": 0,                     //Alarm status (1byte), 0-normal,
    // 1-low voltage alarm, 16-removal alarm, 17-low voltage & removal alarm
           "status": "00000000",              //Tag status (binary string)
        "time": "2018-02-26 15:20:40"     //Tag reading time
      },
         {......},{......}                    //Omitted here
      ]
        }
        @param result_length (Out parameter) specify the length of json_result result
        @return Returns the number of tags; returns a negative number on error (see the definition of error codes)
*/
RFID_DLL_API int RFID_TagDataParse(const unsigned char *recv_buffer, const int recv_len,
                                   char *json_result, int *result_length);



/**
   Convert the tag ID obtained after parsing

   @param tagid Tag ID string
    @param format Specify the format to be converted: 1, Decimal tag  ID format; 2, structured 12-digit tag ID format (type, maker code, batch, item number)
    @param tagid_result(Export parameter) Return the converted tag ID (the length is not less than 13 bytes)
   @return Return 0 on success; return a negative number on error (see error code definition)
*/
RFID_DLL_API int RFID_TagidConvert(const char* tagid, const int format, char *tagid_result);
```