Table of Contents

Smartcard API 1
Structure 1
BS2CSNCard 1
BS2SmartCardHeader 2
BS2SmartCardCredentials 3
BS2AccessOnCardData 3
BS2SmartCardData 3
BS2Card 4

Smartcard API

API that provides a function that reads and writes card data.

- BS2_ScanCard: Scans the card from the device and analyzes it.
- BS2_WriteCard: Writes data to the smart card.
- BS2_EraseCard: Formats the smart card.

Structure

BS2CSNCard

```
typedef struct {
    uint8_t type;
    uint8_t size;
    uint8_t data[BS2_CARD_DATA_SIZE];
} BS2CSNCard;
```

1. *type*
The code value of card type. The card type is to indicate the purpose of the card. When transferring a user from device to server, the Access card will be used only to keep the issue history, since the Access card will work on its own without any user information.

Value	Description	Used Format
0x00	Unknown card	
0x01	CSN card	
0x02	Secure card	
0x03	Access card	
0x0A	Wiegand card	BS2WiegandConfig.format (This format is used when BS2WiegandConfig.CSNIndex and BS2WiegandConfig.CardMask is set as 0
0x1A	Wiegand card	BS2WiegandMultiConfig.formats[0]
0x2A	Wiegand card	BS2WiegandMultiConfig.formats[1]
0x3A	Wiegand card	BS2WiegandMultiConfig.formats[2]
0x4A	Wiegand card	BS2WiegandMultiConfig.formats[3]
0x5A	Wiegand card	BS2WiegandMultiConfig.formats[4]
0x6A	Wiegand card	BS2WiegandMultiConfig.formats[5]
0x7A	Wiegand card	BS2WiegandMultiConfig.formats[6]
0x8A	Wiegand card	BS2WiegandMultiConfig.formats[7]
0x9A	Wiegand card	BS2WiegandMultiConfig.formats[8]
0xAA	Wiegand card	BS2WiegandMultiConfig.formats[9]
0xBA	Wiegand card	BS2WiegandMultiConfig.formats[10]
0xCA	Wiegand card	BS2WiegandMultiConfig.formats[11]
0xDA	Wiegand card	BS2WiegandMultiConfig.formats[12]
0xEA	Wiegand card	BS2WiegandMultiConfig.formats[13]
0xFA	Wiegand card	BS2WiegandMultiConfig.formats[14]

2. *size*
The size of card template.

3. *data*
The data of card template.
In case of Secure Credential Card(SCC), users need to have card information which includes Card ID(24byte), issueCount(4byte) and TimeStamp(4byte).

BS2SmartCardHeader

```
typedef struct {
    uint16_t hdrCRC;
    uint16_t cardCRC;
    BS2_CARD_TYPE cardType;
    uint8_t numOfTemplate;
    uint16_t templateSize;
    uint16_t issueCount;
    uint8_t duressMask;
    uint8_t reserved[5];
} BS2SmartCardHeader;
```

1. *hdrCRC*
Value of card header checksum.

2. *cardCRC*
Value of card data checksum.

3. *cardType*
Code of card types.

Value	Description
0	Unknown card
1	CSN card
2	Secure card
3	Access card
4	Wiegand card

4. *numOfTemplate*
Number of templates.

5. *templateSize*
Size of the template. A normal fingerprint template is a fixed 384 byte.
If you are using a smart card the default in BioStar 2 is 300 bytes and you can change as required but we recommend that you set it above 300 bytes because if the template size is too small it can cause fingerprint matching issues because of the lack of information in the template.

6. *issueCount*
Number of smart card issue count.

7. *duressMask*
Mask for whether there is a duress finger.

BS2SmartCardCredentials

```
typedef struct {
    uint8_t pin[BS2_PIN_HASH_SIZE];
    uint8_t templateData[BS2_SMART_CARD_MAX_TEMPLATE_COUNT *
BS2_FINGER_TEMPLATE_SIZE];
} BS2SmartCardCredentials;
```

1. *pin*
Value of PIN.

2. *templateData*
List of fingerprint template data area, which can be stored up to 4 fingerprint templates.

BS2AccessOnCardData

```
typedef struct {
    uint16_t accessGroupID[BS2_SMART_CARD_MAX_ACCESS_GROUP_COUNT];
    BS2_DATETIME startTime;
    BS2_DATETIME endTime;
} BS2AccessOnCardData;
```

1. *accessGroupID*
List of access group IDs.

2. *startTime*
Starting time where a user can authenticate. When it is set as 0, there are no limits.

3. *endTime*
Ending time where a user can authenticate. When it is set as 0, there are no limits.

BS2SmartCardData

```
typedef struct {
    BS2SmartCardHeader header;
    union {
        uint8_t cardID[BS2_CARD_DATA_SIZE];
        struct {
            uint8_t secureCredentialCardID[BS2_CARD_DATA_SIZE - 8];
            uint32_t issueCount;
            uint32_t issueTimeStamp;
        };
    BS2SmartCardCredentials credentials;
    BS2AccessOnCardData accessOnData;
```

```
    };
} BS2SmartCardData;
```

1. *header*
Smart card header.

2. *cardID*
Card ID that will be used on the card. Access on Cards will need to use the 32 byte array for the card ID.

3. *secureCredentialCardID*
Card ID that will be used on the card and device. Secure Credential Cards will need to use a 24 byte array for the card ID.

4. *issueCount*
The count on how many times the card was issued. This needs to be correctly entered same as the 'issueCount' field from the BS2SmartCardHeader.

5. *issueTimeStamp*
The time when the card was issued. The unit is in Unix timestamp.

6. *credentials*
Authentication data area where the PIN or fingerprint template is stored.

7. *accessOnData*
Data area the AOC card uses, which carries the access group information.

BS2Card

```
typedef struct
{
    uint8_t isSmartCard;
    union
    {
        BS2CSNCard card;
        BS2SmartCardData smartCard;
    };
}BS2Card;
```

1. *isSmartCard*
Decides whether it is a smart card.

2. *card*
CSN card data.

3. *smartCard*
Smart card data.