Table of Contents

Configuration API

The following APIs are used to read and write system configuration information.

- BS2_ResetConfig: Initializes the device's configurations.
- BS2_ResetConfigExceptNetInfo: Initializes the setting information of the device. (Excluding network settings)
- BS2_GetConfig: Retrieves configuration blob from the device.
- BS2_SetConfig: Stores configuration blob on the device.
- BS2_GetFactoryConfig: Retrieves software version and hardware settings from the device.
- BS2_GetSystemConfig: Retrieves system settings from the device.
- BS2_SetSystemConfig: Stores system settings on the device.
- BS2_GetAuthConfig: Retrieves authentication settings from the device.
- BS2_SetAuthConfig: Stores authentication settings on the device.
- BS2_GetStatusConfig: Retrieves LED and buzzer settings from the device.
- BS2_SetStatusConfig: Stores LED and buzzer settings on the device.
- BS2_GetDisplayConfig: Retrieves sound and UI settings from the device.
- BS2_SetDisplayConfig: Stores sound and UI settings on the device.
- BS2_GetIPConfig: Retrieves IP settings from the device.
- BS2_GetIPConfigViaUDP: Retrieves IP settings from the device via the UDP broadcasting.
- BS2_SetIPConfig: Stores IP settings on the device.
- BS2_SetIPConfigViaUDP: Stores IP settings on the device via the UDP broadcasting.
- BS2_GetIPConfigExt: Retrieves DNS and server URL settings from the device.
- BS2_SetIPConfigExt: Stores DNS and server URL settings on the device.
- BS2_GetTNAConfig: Retrieves time and attendance settings from the device.
- BS2_SetTNAConfig: Stores time and attendance settings on the device.
- BS2_GetCardConfig: Retrieves smart card settings from the device.
- BS2_SetCardConfig: Stores smart card settings on the device.
- BS2_GetFingerprintConfig: Retrieves fingerprint matching settings from the device.
- BS2_SetFingerprintConfig: Stores fingerprint matching settings on the device.
- BS2_GetRS485Config: Retrieves RS-485 network settings from the device.
- BS2_SetRS485Config: Stores RS-485 network settings on the device.
- BS2_GetWiegandConfig: Retrieves Wiegand I/O settings from the device.
- BS2_SetWiegandConfig: Stores Wiegand I/O settings on the device.
- BS2_GetWiegandDeviceConfig: Retrieves Wiegand device settings from the device.
- BS2_SetWiegandDeviceConfig: Stores Wiegand device settings on the device.
- BS2_GetInputConfig: Retrieves supervised input port settings from the device.
- BS2_SetInputConfig: Stores supervised input port settings on the device.
- BS2_GetWlanConfig: Retrieves wireless LAN settings from the device.
- BS2_SetWlanConfig: Stores wireless LAN settings on the device.
- BS2_GetTriggerActionConfig: Retrieves trigger and action settings from the device.
- BS2_SetTriggerActionConfig: Stores trigger and action settings on the device.
- BS2_GetEventConfig: Retrieves image log filter settings from the device.
- BS2_SetEventConfig: Stores image log filter settings on the device.
- BS2_GetWiegandMultiConfig: Retrieves Multi-Wiegand settings from the device.
- BS2_SetWiegandMultiConfig: Stores Multi-Wiegand settings on the device.
- BS2_GetCard1xConfig: Retrieves v1 Template on Card settings from the device.
- BS2_SetCard1xConfig: Stores v1 Template on Card settings on the device.
- BS2_GetSystemExtConfig: Retrieves Master and slave device encryption settings from the

device.
- BS2_SetSystemExtConfig: Stores Master and slave device encryption settings on the device
- BS2_GetVoipConfig: Retrieves VoIP settings from the device.
- BS2_SetVoipConfig: Stores VoIP settings on the device.
- BS2_GetFaceConfig: Retrieves face settings from the device.
- BS2_SetFaceConfig: Stores face settings on the device.
- BS2_GetRS485ConfigEx: In case of Corestation, retrieves RS-485 network settings from the device.
- BS2_SetRS485ConfigEx: In case of CoreStation, stores RS-485 network settings on the device.
- BS2_GetCardConfigEx: Retrieves iClass SEOS card settings from the device.
- BS2_SetCardConfigEx: Stores iClass SEOS card settings on the device.
- BS2_GetDstConfig: Retrieves the device DST information.
- BS2_SetDstConfig: Stores the device DST information.
- BS2_GetSupportedConfigMask: Retrieves supported configuration of the device.
- BS2_GetIPConfigViaUDPEx: [+ 2.6.3] Retrieves IP configuration through UDP broadcast with host IP.
- BS2_SetIPConfigViaUDPEx: [+ 2.6.3] Stores IP configuration through UDP broadcast with host IP.
- BS2_GetIPV6Config: [+ 2.6.3] Retrieves IPv6 configuration information.
- BS2_SetIPV6Config: [+ 2.6.3] Stores IPv6 configuration information.
- BS2_GetIPV6ConfigViaUDP: [+ 2.6.3] Retrieves IPv6 configuration through UDP broadcast.
- BS2_SetIPV6ConfigViaUDP: [+ 2.6.3] Stores IPv6 configuration through UDP broadcast.
- BS2_GetIPV6ConfigViaUDPEx: [+ 2.6.3] Retrieves IPv6 configuration through UDP broadcast with host IP.
- BS2_SetIPV6ConfigViaUDPEx: [+ 2.6.3] Stores IPv6 configuration through UDP broadcast with host IP.
- BS2_GetDesFireCardConfigEx: [+ 2.6.4] Retrieves DesFire advanced configuration from the device.
- BS2_SetDesFireCardConfigEx: [+ 2.6.4] Sets DesFire advanced configuration in the device.
- BS2_GetAuthConfigExt: [+ 2.7.1] `FaceStation F2` Retrieves authentication settings from the device.
- BS2_SetAuthConfigExt: [+ 2.7.1] `FaceStation F2` Stores authentication settings from the device.
- BS2_GetFaceConfigExt: [+ 2.7.1] `FaceStation F2, FaceStation2` Retrieves configuration of thermal camera and mask detection.
- BS2_SetFaceConfigExt: [+ 2.7.1] `FaceStation F2, FaceStation2` Stores configuration of thermal camera and mask detection.
- BS2_GetThermalCameraConfig: [+ 2.7.1] `FaceStation F2, FaceStation2` Retrieves configuration of thermal camera.
- BS2_SetThermalCameraConfig: [+ 2.7.1] `FaceStation F2, FaceStation2` Stores configuration of thermal camera.
- BS2_GetBarcodeConfig: [+ 2.8] `X-Station 2` Retrieves configuration of Barcode.
- BS2_SetBarcodeConfig: [+ 2.8] `X-Station 2` Stores configuration of Barcode.
- BS2_GetInputConfigEx: [+ 2.8.1] `IM-120` Retrieves Expanded Configuration related to the Input.
- BS2_SetInputConfigEx: [+ 2.8.1] `IM-120` Retrieves Expanded Configuration related to the Input.
- BS2_GetRelayActionConfig: [+ 2.8.1] `IM-120` Retrieves Configuration related to the RelayAction.
- BS2_SetRelayActionConfig: [+ 2.8.1] `IM-120` Retrieves Configuration related to the RelayAction.

Structure

BS2FactoryConfig

```
typedef struct {
    uint8_t major;
    uint8_t minor;
    uint8_t ext;
    uint8_t reserved[1];
} Version;

typedef struct {
    uint32_t deviceID;
    uint8_t macAddr[BS2_MAC_ADDR_LEN];
    uint8_t reserved[2];
    char modelName[BS2_MODEL_NAME_LEN];
    Version boardVer;
    Version kernelVer;
    Version bscoreVer;
    Version firmwareVer;
    char kernelRev[BS2_KERNEL_REV_LEN];
    char bscoreRev[BS2_BSCORE_REV_LEN];
    char firmwareRev[BS2_FIRMWARE_REV_LEN];
    uint8_t reserved2[32];
} BS2FactoryConfig;
```

1. *deviceID*
Device identifier.

2. *macAddr*
MAC address of the network adaptor.

3. *reserved*
Reserved space.

4. *modelName*
Model name.

5. *boardVer*
Hardware version.

6. *kernelVer*
Kernel version.

7. *bscoreVer*
BioStar Core version.

8. *firmwareVer*
Firmware version.

9. *kernelRev*
Kernel revision information.

10. *bscoreRev*
BioStar Core revision information.

11. *firmwareRev*
Firmware revision information.

12. *reserved2*
Reserved space.

BS2SystemConfig

```
typedef struct {
    uint8_t notUsed[16 * 16 * 3];
    int32_t timezone;
    uint8_t syncTime;
    uint8_t serverSync;
    uint8_t deviceLocked;
    uint8_t useInterphone;
    uint8_t useUSBConnection;
    uint8_t keyEncrypted;
    uint8_t useJobCode;
    uint8_t useAlphanumericID;
    uint32_t cameraFrequency;
    bool secureTamper;
    bool reserved0;     // (write protected)
    uint8_t reserved[2];
    uint32_t useCardOperationMask;
    uint8_t reserved2[16];
} BS2SystemConfig;
```

1. *notUsed*
Not used.

2. *timezone*
Represents standard time zone in seconds.

3. *syncTime*
Stores when synchronization with BioStar has occurred.

4. *serverSync*
Reserved variable.

5. *deviceLocked*
Indicates the current locked state of the device. (Read only filed)

6. *useInterphone*
Decides whether to use intercom.

7. *useUSBConnection*
This is not used anymore. (The device automatically detects USB connection.)

8. *keyEncrypted*
Decides whether to use OSDP secure key.

9. *useJobCode*
Decides whether to use job codes.

10. *useAlphanumericID*
Decides whether to use alphanumeric ID.

11. *cameraFrequency*
Frequency of the camera.

Value	Description
1	50Hz
2	60Hz

12. *secureTamper
Flag to determine whether to use a security tamper.
When Tamper on, the following data is deleted from the device. (User, log, data encryption key, SSL certificate)

13. *reserved0*
Reserved space.

14. *reserved*
Reserved space.

15. *useCardOperationMask*
[+ V2.6.4] Provides a card selective option not to read all kinds of cards from the device.
You can select multiple cards using MASK. The user can select or deselect of a specific card reading option using this option.
However, it can be applied to the card types the device supporting. If you add a card type which isn't supported from the device would be ignored.
Also, the required card type MASK should be combined with `CARD_OPERATION_USE`.
For example, useCardOperationMask needs to be configured `x0x80000001` when EM card is selected only.

Value	Description
0xFFFFFFFF	CARD_OPERATION_MASK_DEFAULT
0x80000000	CARD_OPERATION_USE
0x00000200	CARD_OPERATION_MASK_BLE
0x00000100	CARD_OPERATION_MASK_NFC
0x00000080	CARD_OPERATION_MASK_SEOS
0x00000040	CARD_OPERATION_MASK_SR_SE
0x00000020	CARD_OPERATION_MASK_DESFIRE_EV1
0x00000010	CARD_OPERATION_MASK_CLASSIC_PLUS
0x00000008	CARD_OPERATION_MASK_ICLASS

Value	Description
0x00000004	CARD_OPERATION_MASK_MIFARE_FELICA
0x00000002	CARD_OPERATION_MASK_HIDPROX
0x00000001	CARD_OPERATION_MASK_EM

16. *reserved2*
Reserved space.

BS2AuthConfig

```
typedef struct {
    uint32_t authSchedule[BS2_NUM_OF_AUTH_MODE];
    uint8_t useGlobalAPB;
    uint8_t globalAPBFailAction;
    uint8_t useGroupMatching;
    uint8_t reserved
    uint8_t reserved[28];
    uint8_t usePrivateAuth;
    uint8_t faceDetectionLevel;
    uint8_t useServerMatching;
    uint8_t useFullAccess;
    uint8_t matchTimeout;
    uint8_t authTimeout;
    uint8_t numOperators;
    uint8_t reserved2[1];
    struct {
        char userID[BS2_USER_ID_SIZE];
        uint8_t level;
        uint8_t reserved[3];
    } operators[BS2_MAX_OPERATORS];
} BS2AuthConfig;
```

1. *authSchedule*
Stores schedules for different types of authentication modes.

It has the following meanings in the value of the array.
If the value in the array is greater than 0, the corresponding authentication mode is enabled.
Biometric information in the descriptions below refers to the fingerprint or face depending on the device.

Value	Code	Description
0	BS2_AUTH_MODE_BIOMETRIC_ONLY	Biometric only
1	BS2_AUTH_MODE_BIOMETRIC_PIN	Biometric + PIN
2	BS2_AUTH_MODE_CARD_ONLY	Card only
3	BS2_AUTH_MODE_CARD_BIOMETRIC	Card + Biometric
4	BS2_AUTH_MODE_CARD_PIN	Card + PIN

Value	Code	Description
5	BS2_AUTH_MODE_CARD_BIOMETRIC_OR_PIN	Card + Biometric or PIN
6	BS2_AUTH_MODE_CARD_BIOMETRIC_PIN	Card + Biometric + PIN
7	BS2_AUTH_MODE_ID_BIOMETRIC	ID + Biometric
8	BS2_AUTH_MODE_ID_PIN	ID + PIN
9	BS2_AUTH_MODE_ID_BIOMETRIC_OR_PIN	ID + Biometric or PIN
10	BS2_AUTH_MODE_ID_BIOMETRIC_PIN	ID + Biometric + PIN

2. *useGlobalAPB*
Decides whether to enable global APB zone.

3. *globalAPBFailAction*
Default action that will be executed when the BioStar application cannot determine if the authentication has violated global APB rules.

Value	Description
0	Not use
1	Operate as soft APB
2	Operate as hard APB

4. *useGroupMatching*
Decides whether to use face group matching.

5. *reserved*
Reserved space.

6. *usePrivateAuth*
Decides whether to use private authentication mode.

7. *faceDetectionLevel*
Level of face detection in user authentication. If the detected face level is lower than the configuration, it will be processed as authentication fail.
When set, the camera view according to Normal/Strict is displayed, access is denied if the device doesn't recognize facial image through image log. Default is 0.

Value	Description
0	Face detection not used
1	Normal mode
2	Strict mode

Only valid for A2. Not used with FaceStation2 or FaceLite.

8. *useServerMatching*
Decides whether to perform fingerprint/face matchings on the server.

9. *useFullAccess*
Decides whether to allow full access to all authenticated users regardless to the access group rules.

10. *matchTimeout*
Timeout in seconds for fingerprint/face matching.

11. *authTimeout*
Timeout in seconds for the user authentication response.

12. *numOperators*
The number of operators defining user privileges.

13. *reserved2*
Reserved space.

14. *userID*
User identifier.

15. *level*
Specifies the privilege of the user when accessing to the device's menu.

Value	Description
0	No privilege
1	Administrative privilege
2	Privilege to change the system settings
3	Privilege to change user information

CAUTION
To add an operator, the ***numOperators*** field needs to be set equivalent to the number of operators that will be added.

16. *reserved*
Reserved space.

BS2StatusConfig

```
typedef struct {
    struct {
        uint8_t enabled;
        uint8_t reserved[1];
        uint16_t count;
        BS2LedSignal signal[BS2_LED_SIGNAL_NUM];
```

```
    } led[BS2_DEVICE_STATUS_NUM];
    uint8_t reserved1[32];
    struct {
        uint8_t enabled;
        uint8_t reserved[1];
        uint16_t count;
        BS2BuzzerSignal signal[BS2_BUZZER_SIGNAL_NUM];
    } buzzer[BS2_DEVICE_STATUS_NUM];
    uint8_t configSyncRequired;
    uint8_t reserved2[31];
} BS2StatusConfig;
```

1. *enabled*
Decides whether to use the LED.

2. *reserved*
Reserved space.

3. *count*
Number of LED signal execution count. When it is set as 0, repeats infinitely.

4. *signal*
List of LED signal patterns, which can be configured up to 3 patterns.

5. *reserved1*
Reserved space.

6. *enabled*
Decides whether to use the buzzer.

7. *reserved*
Reserved space.

8. *count*
Number of buzzer signal execution count. When it is set as 0, repeats infinitely.

9. *signal*
List of buzzer signal patterns, which can be configured up to 3 patterns.

10. *configSyncRequired*
If the device's configuration has been modified, this value will be set to true.

11. *reserved2*
Reserved space.

BS2DisplayConfig

```
typedef struct {
```

```
    uint32_t language;
    uint8_t background;
    uint8_t volume;
    uint8_t bgTheme;
    uint8_t dateFormat;
    uint16_t menuTimeout;
    uint16_t msgTimeout;
    uint16_t backlightTimeout;
    uint8_t displayDateTime;
    uint8_t useVoice;
    uint8_t timeFormat;
    uint8_t homeFormation;
    BS2_BOOL useUserPhrase;
    BS2_BOOL queryUserPhrase;
    uint8_t shortcutHome[BS2_MAX_SHORTCUT_HOME];
    uint8_t tnaIcon[16];
    uint8_t useScreenSaver;
    uint8_t reserved1[31];
} BS2DisplayConfig;
```

1. *language*
Language code.

Value	Description
0	Korean
1	English
2	Custom

2. *background*
Background image type.

Value	Description
0	LOGO
1	NOTICE
2	SLIDE
3	PDF

3. *volume*
The volume of sound. The volume can be set from 0 to 100. 0 means no sound.

4. *bgTheme*
Theme type.

Value	Description
0	Logo image
1	Notice
2	Slide show
3	PDF

5. *dateFormat*

Date format.

Value	Description
0	YYYY/MM/DD
1	MM/DD/YYYY
2	DD/MM/YYYY

6. *menuTimeout*
Timeout in seconds for lock screen when the user is inactive. The timeout can be set from 0 to 255 seconds. 0 means no lock screen.

Value	Description
0	No timeout.
10	Menu timeout 10 sec.
20	Menu timeout 20 sec. (Default)
30	Menu timeout 30 sec.
40	Menu timeout 40 sec.
50	Menu timeout 50 sec.
60	Menu timeout 60 sec.

7. *msgTimeout*
Message timeout in milliseconds. The timeout can be set from 500 to 5000 milliseconds.

Value	Description
500	Message timeout 500 msec.
1000	Message timeout 1 sec.
2000	Message timeout 2 sec. (Default)
3000	Message timeout 3 sec.
4000	Message timeout 4 sec.
5000	Message timeout 5 sec.

8. *backlightTimeout*
Backlight timeout in seconds.

Value	Description
0	Backlight timeout 0 sec.
10	Backlight timeout 10 sec.
20	Backlight timeout 20 sec. (Default)
30	Backlight timeout 30 sec.
40	Backlight timeout 40 sec.
50	Backlight timeout 50 sec.
60	Backlight timeout 60 sec.

9. *displayDateTime*
Decides whether to display clock on screen.

10. *useVoice*
Decides whether to use voice instruction.

11. *timeFormat*
Time format.

Value	Description
0	12 hour
1	24 hour

However, Linux OS devices like BioStation 2, BioStation L2, BioLite N2 and FaceLite have opposite settings.(0 = 24 hour / 1 = 12 hour)

12. *homeFormation*
Home screen settings(Currently, not used).

Value	Description
1	Interphone
2	Shortcut 1
3	Shortcut 2
4	Shortcut 3
5	Shortcut 4

13. *useUserPhrase*
Flag that determines whether to use the user phrase feature.

13. *queryUserPhrase*
If set true, asks for the user phrase to the server.

15. *shortcutHome*
Home screen layout(Going to apply later, not used currently).

16. *tnaIcon*
Icon displayed on the device corresponding TNA key.

17. *useScreenSaver*
`FaceStation 2`, `FaceStation F2` If set true, you can activate the screensaver.

18. *reserved1*
Reserved space.

BS2IpConfig

```
typedef struct {
    uint8_t connectionMode;
    uint8_t useDHCP;
    uint8_t useDNS;
    uint8_t reserved[1];
    char ipAddress[BS2_IPV4_ADDR_SIZE];
    char gateway[BS2_IPV4_ADDR_SIZE];
    char subnetMask[BS2_IPV4_ADDR_SIZE];
```

```
    char serverAddr[BS2_IPV4_ADDR_SIZE];
    uint16_t port;
    uint16_t serverPort;
    uint16_t mtuSize;
    uint8_t baseband;
    uint8_t reserved2[1];
    uint16_t sslServerPort;
    uint8_t reserved3[30];
} BS2IpConfig;
```

1. *connectionMode*
Represents the connection mode between BioStar and devices. There are two modes depending on who initiates the connection: Direct mode (0x0) and Server mode (0x1). The Direct mode means that BioStar initiates the connection to the devices and the Server mode means that the devices initiate the connection to the server. The default connection mode of a device is the Direct mode.

2. *useDHCP*
Decides whether to use DHCP.

3. *useDNS*
Decides whether to use server address or server URL.

4. *reserved*
Reserved space.

5. *ipAddress*
IP address assigned to the device.

6. *gateway*
IP address of the gateway.

7. *subnetMask*
Subnet mask of the device.

8. *serverAddr*
IP address of BioStar. Used only in the server mode.

9. *port*
Port number of the device.

10. *serverPort*
Port number of BioStar. Used only in the server mode.

11. *mtuSize*
MTU[1)] size for the TCP/IP communication.

12. *baseband*
Bandwidth of the device. The value can be set to 10MB/S or 100 MB/S.

13. *reserved2*
Reserved space.

14. *sslServerPort*
Used when the connectionMode is set as server SSL mode, which is the port of the SDK application.

15. *reserved3*
Reserved space.

BS2IpConfigExt

```
typedef struct {
    char dnsAddr[BS2_IPV4_ADDR_SIZE];
    char serverUrl[BS2_URL_SIZE];
    uint8_t reserved[32];
} BS2IpConfigExt;
```

1. *dnsAddr*
DNS server address.

2. *serverUrl*
URL of the BioStar application server. The maximum length is 256 characters.

3. *reserved*
Reserved space.

BS2TNAConfig

```
typedef struct {
    uint8_t tnaMode;
    uint8_t tnaKey;
    uint8_t tnaRequired;
    uint8_t reserved[1];
    uint32_t tnaSchedule[BS2_MAX_TNA_KEY];
    uint8_t unused[BS2_MAX_TNA_KEY];
} BS2TNAInfo;

typedef struct {
    char tnaLabel[BS2_MAX_TNA_KEY][BS2_MAX_TNA_LABEL_LEN];
    uint8_t unused[BS2_MAX_TNA_KEY];
} BS2TNAExtInfo;

typedef struct {
    BS2TNAInfo tnaInfo;
    BS2TNAExtInfo tnaExtInfo;
    uint8_t reserved2[32];
} BS2TNAConfig;
```

1. *tnaMode*
Time and attendance management mode.

Value	Description
0	Not used
1	Applying time and attendance code according to a user
2	Applying time and attendance code according to a T&A schedule
3	Applying the time and attendance code that the previous user has selected
4	Using a fixed time and attendance code

2. *tnaKey*
Represents time and attendance code. This is mapped to a key on the device.

Device Type	T&A Code	Mapped Key	Value
BioStation 2	BS2_TNA_UNSPECIFIED	(N/A)	0
	BS2_TNA_KEY_1	F1	1
	BS2_TNA_KEY_2	F2	2
	BS2_TNA_KEY_3	F3	3
	BS2_TNA_KEY_4	F4	4
	BS2_TNA_KEY_5	1	5
	BS2_TNA_KEY_6	2	6
	BS2_TNA_KEY_7	3	7
	BS2_TNA_KEY_8	4	8
	BS2_TNA_KEY_9	5	9
	BS2_TNA_KEY_10	6	10
	BS2_TNA_KEY_11	7	11
	BS2_TNA_KEY_12	8	12
	BS2_TNA_KEY_13	9	13
	BS2_TNA_KEY_14	Call	14
	BS2_TNA_KEY_15	0	15
	BS2_TNA_KEY_16	Esc	16

3. *tnaRequired*
Decides whether to time and attendance code entry is mandatory when the time and attendance management mode is set to 1.

4. *reserved*
Reserved space.

5. *tnaSchedule*
Specifies a schedule for a time and attendance code.

6. *unused*
Not used.

7. *tnaLabel*
A label that shows the meaning of the time and attendance code.

8. *unused*
Not used.

BS2CardConfig

```
typedef struct {
    uint8_t primaryKey[6];
    uint8_t reserved1[2];
    uint8_t secondaryKey[6];
    uint8_t reserved2[2];
    uint16_t startBlockIndex;
    uint8_t reserved[6];
} BS2MifareCard;

typedef struct {
    uint8_t primaryKey[8];
    uint8_t secondaryKey[8];
    uint16_t startBlockIndex;
    uint8_t reserved[6];
} BS2IClassCard;

typedef struct {
    uint8_t primaryKey[16];
    uint8_t secondaryKey[16];
    uint8_t appID[3];
    uint8_t fileID;
    uint8_t encryptionType;
    uint8_t operationMode;
    uint8_t reserved[2];
} BS2DesFireCard;

typedef struct {
    uint8_t byteOrder;
    uint8_t useWiegandFormat;
    uint8_t dataType;
    uint8_t useSecondaryKey;
    BS2MifareCard mifare;
    BS2IClassCard iclass;
    BS2DesFireCard desfire;
    uint8_t formatID;
    uint8_t cipher;
    uint8_t reserved[24];
} BS2CardConfig;
```

1. *primaryKey*
Primary encryption key to access the Mifare card information.

2. *reserved1*
Reserved space.

3. *secondaryKey*
Secondary encryption key to access the Mifare card information.

4. *reserved2*
Reserved space.

5. *startBlockIndex*
Start block index on the Mifare data storage.

6. *reserved*
Reserved space.

7. *primaryKey*
Primary encryption key to access the iClass card information.

8. *secondaryKey*
Secondary encryption key to access the iClass card information..

9. *startBlockIndex*
Start block index on the Mifare data storage.

10. *reserved*
Reserved space.

11. *primaryKey*
Primary encryption key to access the DesFire card information.

12. *secondaryKey*
Secondary encryption key to access the Desfire card information.

13. *appID*
Application Id that is stored inside the DesFire card for user authentication.

14. *fileID*
File ID that is stored inside the DesFire card, which will be used by the application to read and write data.

15. *encryptionType*
Type of data encryption.

Value	Description
0	DES/3DES
1	AES

16. *operationMode*
Operation mode. (operationMode will be supported soon.)

Value	Description
0	Lagacy mode (Using PICC master key)
1	New mode (Using App master key)

17. *reserved*
Reserved space.

18. *byteOrder*
Order of how the byte of the card is stored. When it is set as 0, will function as MSB[2]. When it is set as 1, will function as LSB[3].

19. *useWiegandFormat*
Decides whether to use Wiegand format.

20. *dataType*
Type of card data.

Value	Description
0	Binary
1	ASCII
2	UTF16
3	BCD

21. *useSecondaryKey*
Decides whether to use the secondary encryption key.

22. *formatID*
ID that is used when the card configuration needs to be managed from the database on the BioStar application.

23. *cipher*
Activates 'Keypad card ID' option.
Default value is 0, it is only valid for Gangbox Keypad type of Xpass 2, XPass D2.

Value	Description
0	Deactivate
1	Activate

24. *reserved5*
Reserved space.

BS2FingerprintConfig

```
typedef struct {
    uint8_t      securityLevel;
    uint8_t      fastMode;
    uint8_t      sensitivity;
    uint8_t      sensorMode;
    uint16_t     templateFormat;
    uint16_t     scanTimeout;
    uint8_t      successiveScan;
    uint8_t      advancedEnrollment;
    uint8_t      showImage;
```

```
    uint8_t      lfdLevel;
    bool         checkDuplicate;

    uint8_t      reserved3[31];
} BS2FingerprintConfig;
```

1. *securityLevel*
Fingerprint authentication security level. This is used across the system.

Value	Description
0	Basic
1	Highly secure
2	The most highly secure

2. *fastMode*
Fingerprint matching speed.

Value	Description
0	Automatic
1	Basic
2	High
3	Very High

3. *sensitivity*
Sensitivity of the fingerprint sensor.

Value	Description
0	Lowest
1	Level 1
2	Level 2
3	Level 3
4	Level 4
5	Level 5
6	Level 6
7	Highest

4. *sensorMode*
Decides the sensor mode. 0 means the sensor is always on. 1 means the sensor is activated when the finger is near the sensor.

5. *templateFormat*
Fingerprint template type.

Value	Description
0	Suprema
1	ISO
2	ANSI

6. *scanTimeout*
Fingerprint scanning timeout in seconds. The default is 10 seconds.

7. *successiveScan*
Not Used.

8. *advancedEnrollment*
Decides whether to utilize fingerprint quality information. If the option is disabled, the BS_SDK_ERROR_CAPTURE_LOW_QUALITY / BS_SDK_ERROR_EXTRACTION_LOW_QUALITY error codes are not returned even if the quality of fingerprint image acquired is low.

9. *showImage*
Decides whether to display scanned fingerprint image on the screen.

10. *lfdLevel*
Configuration for the LFD(Live Fingerprint Detection - fake fingerprint detection) sensitivity.

Value	Description
0	Not Use
1	Strict
2	More Strict
3	Most Strict

11. *checkDuplicate*
[+ V2.6.4] If set to true, it will determine if the fingerprint is a duplicate.

12. *reserved3*
Reserved space.

BS2Rs485Config

```
typedef struct {
    uint8_t supportConfig;
    uint8_t useExceptionCode;
    uint8_t exceptionCode[BS2_RS485_MAX_FAIL_CODE_LEN];
    uint8_t outputFormat;
    uint8_t osdpID;
    uint8_t reserved[4];
} BS2IntelligentPDInfo;   //Added 2.8.0 for Intelligent Slave Feature

typedef struct {
    uint32_t baudRate;
    uint8_t channelIndex;
    uint8_t useRegistance;
    uint8_t numOfDevices;
    uint8_t reserved[1];
    BS2Rs485SlaveDevice slaveDevices[BS2_RS485_MAX_SLAVES_PER_CHANNEL];
} BS2Rs485Channel;

typedef struct {
    uint8_t mode;
```

```
    uint8_t numOfChannels;
    uint8_t reserved[2];
    BS2IntelligentPDInfo intelligentInfo;  //Updated to v2.8.0
    uint8_t reserved1[16];
    BS2Rs485Channel channels[BS2_RS485_MAX_CHANNELS];
} BS2Rs485Config;
```

1. *supportConfig*
[+V2.8] If this value is 0, the device will ignore all settings related to Intelligent PD (Peripheral Device) below.
useExceptionCode
exceptionCode
outputFormat
osdpID

2. *useExceptionCode*
[+V2.8] This option is available to choose whether the exception code is sent or not.

3. *exceptionCode*
[+V2.8] This function sends an exception code in case of authentication failure or authentication success but no card registered user.
Set the exception code to be used at this time.
If the exception code is 0 (0x0000000000000000), no exception code is generated.

4. *outputFormat*
[+V2.8] Intelligent Slave device can send Card ID or User ID upon successful authentication.
If it is 0, the card ID is output, if it is 1, the user ID is output.

5. *osdpID*
[+V2.8] This is a value used to distinguish two or more Suprema Intelligent devices from each other when connecting to the RS485 port of the same third party controller. You can set and designate a unique value between 0 and 127. The default OSDP ID for Suprema intelligent devices is 0.

6. *reserved*
[+V2.8] Reserved space.

7. *baudRate*
The RS-485 communication speed which can be configured as below.

Value
9600
19200
38400
57600
115200

8. *channelIndex*
(non configurable index) Communication channel index of the RS-485 network.

9. *useRegistance*
Registance flag - no effect on operation.

10. *numOfDevices*
Number of slave devices.

11. *reserved*
Reserved space.

12. *slaveDevices*
List of slave devices, which can be configured up to 32 devices.

13. *mode*
Decides the operating mode on the RS-485 network.

Value	Description
0	Not use
1	Master
2	Slave
3	Standalone

14. *numOfChannels*
Number of RS-485 channel.

15. *reserved*
Reserved space.

16. *intelligentInfo*
`[+V2.8]` This is Intelligent Slave Device Information
This only works when the device mode is RS485 default.
Once the Suprema device is connected to a slave device to the 3rd party ACU through RS485(OSDP), the Suprema device becomes a Peripheral Device of the OSDP automatically.

17. *reserved1*
Reserved space.

18. *channels*
List of RS-485 channels, which can be configured up to 4 channels.

BS2WiegandConfig

```
typedef struct {
    uint32_t length;
    uint8_t idFields[BS2_WIEGAND_MAX_FIELDS][BS2_WIEGAND_FIELD_SIZE];
    uint8_t parityFields[BS2_WIEGAND_MAX_PARITIES][BS2_WIEGAND_FIELD_SIZE];
    BS2_WIEGAND_PARITY parityType[BS2_WIEGAND_MAX_PARITIES];
    uint8_t parityPos[BS2_WIEGAND_MAX_PARITIES];
} BS2WiegandFormat;

typedef struct {
    uint8_t mode;
```

```
    uint8_t useWiegandBypass;
    uint8_t useFailCode;
    uint8_t failCode;
    uint16_t outPulseWidth;
    uint16_t outPulseInterval;
    uint32_t formatID;
    BS2WiegandFormat format;
    uint16_t wiegandInputMask;
    uint16_t wiegandCardMask;
    uint8_t wiegandCSNIndex;
    uint8_t useWiegandUserID;
    uint8_t reserved[26];
} BS2WiegandConfig;
```

1. *length*
The length of the wiegand card format.

2. *idFields*
You can set 4 id fields maximum. Each field's id needs to be inserted from the beginning to the end starting from the end of the array. For example, Standard 26bit wiegand card data is made up as “P FFFFFFFF NNNNNNNNNNNNNNNN P”. The Facility Code is “0 11111111 0000000000000000 0”, so it has the value of 0x01FE0000, and the Card Number has the value of 0x0001FFFE.

```
// for Facillity Code
idFields[][28] = 0x01;
idFields[][29] = 0xFE;
idFields[][30] = 0x00;
idFields[][31] = 0x00;

// for Card Number
idFields[1][28] = 0x00;
idFields[1][29] = 0x01;
idFields[1][30] = 0xFF;
idFields[1][31] = 0xFE;
```

3. *parityFields*
You can set 4 parity fields maximum, and enter the beginning and the end of the range where to check the parity.

4. *parityType*
Set the parity type.

Value	Description
0	Does not check parity
1	check odd parity
2	check even parity

5. *parityPos*
Select the position of the parity bit on the wiegand card data.

6. *mode*

Set the wiegand Input/Output mode.

Value	Description
0	Input
1	Output
2	In/Output

7. *useWiegandBypass*
The flag that indicates whether to send out a card data.

Value	Description
0	Output when authenticated.
1	Output without authentication.

8. *useFailCode*
The flag that indicates whether to send out a fail code when a card fails to authenticate.

9. *failCode*
Fail code value to replace the card data.

Value
0x00
0xFF

10. *outPulseWidth*
Output pulse width having a range of 20 ~ 100 us.

11. *outPulseInterval*
Output pulse frequency having a range of 200 ~ 20000 us.

12. *formatID*
The value used to distinguish the wiegand card format in the application, and this is not used from the device.

13. *format*
Wiegand format structure.

14. *wiegandInputMask*
Input mask for the wiegand input of the slave and wiegand device.

15. *wiegandCardMask*
Input mask for the wiegand input of the master device.

16. *wiegandCSNIndex*
Index that determines in which format the device will send out a wiegand ouput after the card has been read. This field is used only for Mifare and EM devices. Please check the *useWiegandFormat* field of the BS2CardConfig structure before configuring.

17. *useWiegandUserID*
The flag you can select whether card ID or user ID to be sent via Wiegand output.

Value	Description
0	Not use
1	Card ID
2	User ID

18. *reserved*
Reserved space.

BS2WiegandDeviceConfig

```
typedef struct {
    uint32_t deviceID;
    uint16_t port;
    uint8_t switchType;
    uint8_t reserved[1];
} BS2WiegandTamperInput;

typedef struct {
    uint32_t deviceID;
    uint16_t port;
    uint8_t reserved[10];
} BS2WiegandLedOutput;

typedef struct {
    uint32_t deviceID;
    uint16_t port;
    uint8_t reserved[34];
} BS2WiegandBuzzerOutput;

typedef struct {
    BS2WiegandTamperInput tamper;
    BS2WiegandLedOutput led[BS2_WIEGAND_STATUS_NUM];
    BS2WiegandBuzzerOutput buzzer;
    uint32_t reserved[32];
} BS2WiegandDeviceConfig;
```

1. *deviceID*
ID of the device which will receive the tamper signal of the Wiegand card reader.

2. *port*
Input port for the Wiegand card reader's tamper.

3. *switchType*
If the switch type is normally open, and the input isgnal is on, it will set off the trigger.

Value	Description
0	Normally open
1	Normally closed

4. *reserved*
Reserved space.

5. *deviceID*
ID of the device which will send the LED signal to the Wiegand card reader.

6. *port*
Output port for the Wiegand card reader's LED signal.

7. *reserved*
Reserved space.

8. *deviceID*
ID of the device which will send the buzzer signal to the Wiegand card reader.

9. *port*
Output port for the Wiegand card reader's buzzer signal.

10. *reserved*
Reserved space.

10. *led*
List of devices sending LED signals of the Wiegand card reader, which can be configured up to 2 devices.

Value	Description
0	Red LED
1	Green LED

BS2InputConfig

```
typedef struct {
    uint16_t minValue;
    uint16_t maxValue;
} BS2SVInputRange;

typedef struct {
    uint32_t deviceID;
    uint16_t port;
    uint8_t reserved[10];
} BS2WiegandLedOutput;

typedef struct {
    BS2SVInputRange shortInput;
    BS2SVInputRange openInput;
    BS2SVInputRange onInput;
    BS2SVInputRange offInput;
} BS2SupervisedInputConfig;

typedef struct {
```

```
    uint8_t numInputs;
    uint8_t numSupervised;
    uint16_t reseved;
    struct {
        uint8_t portIndex;
        uint8_t enabled;
        uint8_t supervised_index;
        uint8_t reserved[5];
        BS2SupervisedInputConfig config;
    } supervised_inputs[BS2_MAX_INPUT_NUM];
} BS2InputConfig;
```

1. *minValue*
Minimum voltage which has a range from 0 ~ 3300(3.3v).

2. *maxValue*
Maximum voltage which has a range from 0 ~ 3300(3.3v).

3. *shortInput*
Range of the voltage which will be distinguished as short input.

4. *openInput*
Range of the voltage which will be distinguished as open input.

5. *onInput*
Range of the voltage which will be distinguished as on input.

6. *offInput*
Range of the voltage which will be distinguished as off input.

7. *numInputs*
Number of input ports.

8. *numSupervised*
Number of the supervised input ports.

9. *portIndex*
Input port number.

10. *enabled*
Decides whether to use as a supervised input.

11. *supervised_index*
Type of supervised input's resistance value.

Value	Description
0	1k resistance
1	2.2k resistance
2	4.7k resistance
3	10k resistance

Value	Description
255	Custom

12. *reserved*
Reserved space.

13. *config*
Configuration that distinguishes the supervised input signal type. This configuration will be valid only when the supervised input's resistance is set as custom .

BS2WlanConfig

```
typedef struct {
    uint8_t enabled;
    uint8_t operationMode;
    uint8_t authType;
    uint8_t encryptionType;
    char essid[BS2_WLAN_SSID_SIZE];
    char authKey[BS2_WLAN_KEY_SIZE];
    uint8_t reserved2[32];
} BS2WlanConfig;
```

1. *enabled*
Decides whether to use the wireless LAN.

2. *operationMode*
Type of wireless LAN.

Value	Description
0	infrastructure
1	Ad-hoc

3. *authType*
Type of Wireless LAN authentication.

Value	Description
0	Open
1	Shared
2	WPA-PSK
3	WPA2-PSK

4. *encryptionType*
Type of wireless LAN encryption.

Value	Description
0	None
1	WEP
2	TKIP/AES
3	AES

Value	Description
3	TKIP

5. *essid*
ESS ID of the wireless network.

6. *authKey*
Password of the wireless network.

7. *reserved*
Reserved space.

BS2Trigger

```
typedef struct {
    uint16_t code;
    uint8_t reserved[2];
} BS2EventTrigger;

typedef struct {
    uint8_t port;
    uint8_t switchType;
    uint16_t duration;
    uint32_t scheduleID;
} BS2InputTrigger;

typedef struct {
    uint32_t type;
    uint32_t scheduleID;
} BS2ScheduleTrigger;

typedef struct {
    uint32_t deviceID;
    uint8_t type;
    uint8_t reserved[3];

    union {
        BS2EventTrigger event;
        BS2InputTrigger input;
        BS2ScheduleTrigger schedule;
    }
} BS2Trigger;
```

1. *code*
Event log that will set off the trigger.

2. *reserved*
Reserved space.

3. *port*
Input port number that will set off the trigger.

4. *switchType*
If the switch type is normally open, and the input isgnal is on, it will set off the trigger.

Value	Description
0	Normally open
1	Normally closed

5. *duration*
The duration time of the signal that will set off the trigger. The unit of time is milliseconds and the minimum value is 100.

6. *scheduleID*
ID of the time schedule when to operate the trigger.

7. *type*
Type of the schedule trigger.

Value	Description
0	Start schedule trigger
1	End schedule trigger

8. *scheduleID*
ID of the time schedule when to operate the trigger.

9. *deviceID*
ID of the device that will perform the trigger.

10. *type*
Type of trigger.

Value	Description
0	None
1	Event trigger
2	Input trigger
3	Schedule trigger

BS2Action

```
typedef struct {
    uint32_t signalID;
    uint16_t count;
    uint16_t onDuration;
    uint16_t offDuration;
    uint16_t delay;
} BS2Signal;

typedef struct {
```

```
    uint8_t portIndex;
    uint8_t reserved[3];
    BS2Signal signal;
} BS2OutputPortAction;

typedef struct {
    uint8_t relayIndex;
    uint8_t reserved[3];
    BS2Signal signal;
} BS2RelayAction;

typedef struct {
    uint8_t color;
    uint8_t reserved[1];
    uint16_t duration;
    uint16_t delay;
} BS2LedSignal;

typedef struct {
    uint16_t count;
    uint8_t reserved[2];
    BS2LedSignal signal[3];
} BS2LedAction;

typedef struct {
    uint8_t tone;
    uint8_t fadeout;
    uint16_t duration;
    uint16_t delay;
} BS2BuzzerSignal;

typedef struct {
    uint16_t count;
    uint8_t reserved[2];
    BS2BuzzerSignal signal[3];
} BS2BuzzerAction;

typedef struct {
    uint8_t duration;
    uint8_t reserved[3];
    uint32_t displayID;
    uint32_t resourceID;
} BS2DisplayAction;

typedef struct {
    uint8_t count;
    uint16_t soundIndex;
    uint8_t reserved[5];
} BS2SoundAction;

typedef struct {
```

```
    uint32_t deviceID;
    uint8_t type;
    uint8_t stopFlag;
    uint16_t delay;
    union {
        BS2RelayAction relay;
        BS2OutputPortAction outputPort;
        BS2DisplayAction display;
        BS2SoundAction sound;
        BS2LedAction led;
        BS2BuzzerAction buzzer;
    };
} BS2Action;
```

1. *signalID*
Index that is used to manage the signal type from the application.

2. *count*
Number of signal execution count.

3. *onDuration*
Duration of the ON signal. The unit of time is milliseconds.

4. *offDuration*
Duration of the OFF signal. The unit of time is milliseconds.

5. *delay*
Delay time before the signal starts. The unit of time is milliseconds/ For example, count(2), onDuration(100), offDuration(100), delay(50) will execute a signal as below.

50ms wait	signal on(100)	signal off(100)	signal on(100)	signal off(100)

6. *portIndex*
Number of the TTL output port.

7. *reserved*
Reserved Space.

8. *relayIndex*
Number of the TTL output port.

9. *reserved*
Reserved Space.

10. *color*
LED colory type.

Value	Description
0	LED Off
1	Red LED
2	Yellow LED

Value	Description
3	Green LED
4	Blue-Green LED
5	Blue LED
6	Magenta LED
7	White LED

11. *reserved*
Reserved space.

12. *duration*
Duration of the LED. The unit of time is milliseconds.

13. *delay*
Delay before the LED flickers. The unit of time is milliseconds.

14. *count*
Number of LED signal count. When set as 0 it is disabled, and when set as -1 it will repeat infinitely.

15. *reserved*
Reserved space.

16. *tone*
Volume of the buzzer.

Value	Description
0	No sound
1	Low
2	Medium
3	High

17. *count*
Number of buzzer signal count. When set as 0 it is disabled, and when set as -1 it will repeat infinitely.

18. *reserved*
Reserved space.

19. *duration*
Duration of the display operation. The unit of time is milliseconds.

20. *reserved*
Reserved space.

21. *displayID*
Not supported yet.

22. *resourceID*
Not supported yet.

23. *count*

Number of the sound signal count.

24. *soundIndex*
Index of the sound resource.

Value	Description
0	Welcome sound
1	Auth success sound
2	Auth fail sound

25. *deviceID*
ID of the device that will execute the action.

26. *type*
Action types.

[DoorModule-20, CoreStation-40]
If the action type is relay or TTL (Output) and the action device is DM20, CS40, Action type should be set only as relay action (6). (TTL setting not possible)

[DM20]

- Action type : Relay
- relay.relayIndex : 0 ~ 3 (RELAY 0 ~ 3)
- relay.relayIndex : 4 ~ 9 (OUTPUT 0 ~ 5)

[CS40]

- Action type : Relay
- relay.relayIndex : 0 ~ 3 (RELAY 0 ~ 3)
- relay.relayIndex : 4 ~ 11 (OUTPUT 0 ~ 7)

Value	Description
0	None
1	Lock device
2	Unlock device
3	Reboot device
4	Release alarm
5	General input
6	Relay action
7	TTL action
8	Sound action
9	Display action
10	Buzzer action
11	Led action
12	Fire alarm input

Value	Description
13	Auth Success(Access granted)
14	Auth Fail(Access denied)
15	Lift action

27. *stopFlag*
Specifies the condition to stop the Action.
If this value is set to 1 and the signal is detected through the door sensor, the action will stop.
If this value is set to 2, it can be stopped only by the current action information.
In general, related APIs that stop an action are called with a zone id, in which case all devices in the zone will stop the action.
By setting stopFlag to 2 and and loading action information, you can selectively control only the alarms of that device.

Value	Description
0	Don't stop
1	Stop if door is closed
2	Stop by command(Added in V2.6.0)

28. *delay*
Action delay. Unit is millisecond(ms).

BS2TriggerActionConfig

```
typedef struct {
    uint8_t numItems;
    uint8_t reserved[3];
    BS2TriggerAction items[BS2_MAX_TRIGGER_ACTION];
    uint8_t reserved2[32];
} BS2TriggerActionConfig;
```

1. *numItems*
Number of trigger actions.

2. *reserved*
Reserved space.

3. *items*
List of trigger actions, which can be configured up to 128 trigger actions.

4. *reserved2*
Reserved space.

BS2EventConfig

```
typedef struct {
    uint32_t numImageEventFilter;
```

```
    struct {
        uint8_t mainEventCode;
        uint8_t reserved[3];
        uint32_t scheduleID;
    } imageEventFilter[BS2_EVENT_MAX_IMAGE_CODE_COUNT];
    uint8_t reserved[32];
} BS2EventConfig;
```

1. *numImageEventFilter*
Number of image log filters.

2. *mainEventCode*
Main code of the log where the image log will be placed.

3. *reserved*
Reserved space.

4. *scheduleID*
ID of the time schedule when to store the image logs.

5. *reserved*
Reserved space.

BS2WiegandMultiConfig

```
typedef struct {
    uint32_t formatID;
    BS2WiegandFormat format;
    uint8_t reserved[32];
} BS2WiegandInConfig;

typedef struct {
    BS2WiegandInConfig formats[MAX_WIEGAND_IN_COUNT];
    uint8_t reserved[32];
} BS2WiegandMultiConfig;
```

1. *formatID*
Wiegand format index.

2. *format*
Wiegand format structure.

3. *reserved*
Reserved space.

4. *formats*
Available to configure up to 15 formats.

5. *reserved*
Reserved space.

BS1CardConfig

```
typedef struct {
    enum {
        MIFARE_KEY_SIZE = 6,
        MIFARE_MAX_TEMPLATE = 4,

        VALID_MAGIC_NO = 0x1f1f1f1f,
    };

    // Options
    uint32_t     magicNo;
    uint32_t     disabled;
    uint32_t     useCSNOnly;         // default 0
    uint32_t     bioentryCompatible; // default 0

    // Keys
    uint32_t    useSecondaryKey;
    uint32_t     reserved1;
    uint8_t     primaryKey[MIFARE_KEY_SIZE];
    uint8_t        reserved2[2];
    uint8_t     secondaryKey[MIFARE_KEY_SIZE];
    uint8_t        reserved3[2];

    // Layout
    uint32_t     cisIndex;
    uint32_t     numOfTemplate;
    uint32_t     templateSize;
    uint32_t     templateStartBlock[MIFARE_MAX_TEMPLATE];

    uint32_t     reserve4[15];

} BS1CardConfig;
```

1. *magicNo*
Not used.

2. *disabled*
Not used.

3. *useCSNOnly*
Flag that indicates whether to read the v1 ToC cards.

4. *bioentryCompatible*
Not used.

5. *useSecondaryKey*
Not used.

6. *reserved1*

Reserved space.

7. *primaryKey*
Not used.

8. *reserved2*
Reserved space.

9. *secondaryKey*
Not used.

10. *reserved3*
Reserved Space.

11. *cisIndex*
Not used.

12. *numOfTemplate*
Number of template that is used.

13. *templateSize*
Size of each template.

14. *templateStartBlock*
Not used.

15. *reserve4*
Reserved Space.

BS2SystemConfigExt

```
typedef struct {
    uint8_t primarySecureKey[SEC_KEY_SIZE];
    uint8_t secondarySecureKey[SEC_KEY_SIZE];

    uint8_t reserved3[32];
} BS2SystemConfigExt;
```

1. *primarySecureKey*
Primary encryption key used between master and slave devices.

2. *secondarySecureKey*
Secondary encryption key used between master and slave devices.

3. *reserved3*
Reserved space.

BS2VoipConfig

```
typedef struct {
    BS2_URL            serverUrl;            ///
    BS2_PORT        serverPort;            ///
    BS2_USER_ID        userID;                ///
    BS2_USER_ID        userPW;                ///

    uint8_t            exitButton;            /// << *, #, 0~9
    uint8_t            dtmfMode;            ///
    BS2_BOOL        bUse;                ///
    uint8_t            reseverd[1];        ///

        uint32_t                numPhonBook;
    BS2UserPhoneItem     phonebook[BS2_VOIP_MAX_PHONEBOOK];  ///

    uint8_t                reserved2[32];        ///
} BS2VoipConfig;
```

1. *serverUrl*
URL of the SIP server.

2. *serverPort*
Port number of the SIP server.

3. *userID*
User ID to access the SIP server.

4. *userPW*
Password that is used to access the SIP server.

5. *exitButton*
Button to be used as an exit button. (*, #, 0~9)

Value	Description
0	*
1	#
2 ~ 11	0 ~ 9

6. *dtmfMode*
Tone of the keypad.

7. *bUse*
Flag that determines whether the VoIP feature is used.

8. *reseverd*
Reserved space.

9. *numPhonBook*

Number of phone books.

10. *phonebook*
List of extension numbers, which can be configured up to 32.

8. *reserved2*
Reserved space.

BS2FaceConfig

```
typedef struct {
    uint8_t       securityLevel;
    uint8_t       lightCondition;
    uint8_t       enrollThreshold;
    uint8_t       detectSensitivity;

    uint16_t      enrollTimeout;
    uint8_t       lfdLevel;
    bool          quickEnrollment;

    uint8_t       previewOption;
    bool          checkDuplicate;
    uint8_t       operationMode;
    uint8_t       maxRotation;

    struct {
        uint16_t  min;
        uint16_t  max;
    } faceWidth;

    struct {
        uint16_t  x;
        uint16_t  width;
    } searchRange;

    uint8_t       reserved2[18];
} BS2FaceConfig;
```

1. *securityLevel*
Face authentication security level. This is used across the system. .

Value	Description
0	Basic
1	Highly secure
2	Most highly secure

2. *lightCondition*
Configuration of the light condition.

Value	Description
0	Indoor
1	Outdoor
2	Automatic
3	[+V2.8] Not used (FaceStation F2 v1.1.0 or higher version)

[Note]
FaceStation F2: v1.0.0 - v1.0.5
Ambient Brightness: Normal, High, Auto
FaceStation F2: v1.1.0 or higher version
Light Brightness: Normal, High, Not Used

3. *enrollThreshold*
Threshold of face enrollment. It determines how much movement of pose is allowed when enrolling the face.

Value	Description
0	THRESHOLD_0 (Most strict)
1	THRESHOLD_1
2	THRESHOLD_2
3	THRESHOLD_3
4	THRESHOLD_4 (Default)
5	THRESHOLD_5
6	THRESHOLD_6
7	THRESHOLD_7
8	THRESHOLD_8
9	THRESHOLD_9 (Least strict)

4. *detectSensitivity*
Configuration of sensitivity on detecting the face.

Value	Description
0	Off
1	Low
2	Medium
3	High

5. *enrollTimeout*
FaceStation2, FaceLite : Timeout period of face scanning which is 60 seconds by default.

Value	Description
BS2_FACE_ENROLL_TIMEOUT_MIN	30
BS2_FACE_ENROLL_TIMEOUT_MAX	60
BS2_FACE_ENROLL_TIMEOUT_DEFAULT	BS2_FACE_ENROLL_TIMEOUT_MAX

FaceStation F2 : [+ V2.7.1] Face scan wait time, default is 20 seconds.

Value	Description
BS2_FACE_EX_ENROLL_TIMEOUT_MIN	10
BS2_FACE_EX_ENROLL_TIMEOUT_MAX	20
BS2_FACE_EX_ENROLL_TIMEOUT_DEFAULT	BS2_FACE_EX_ENROLL_TIMEOUT_MAX

6. *lfdLevel*

[+ V2.6.3] Configuration for the LFD(Live Face Detection - fake face detection) sensitivity.
`FaceStation2, FaceLite` : Default is 0.
`FaceStation F2` : [+ V2.7.1] Default is 1.

Value	Description
0	Not Use
1	Strict
2	More Strict
3	Most Strict

7. *quickEnrollment*

[+ V2.6.3] Quick face enrollment process.
True - Face enrollment process with a single step.
False - Face enrollment process with 3 steps.
Please use false if you want to enroll with a high quality of face templates.

8. *previewOption*

[+ V2.6.3] IR camera preview option when you authenticate with the face.
Only used to `FaceLite`.

Value	Description
0	Preview not used
1	Preview not used at first of authentication, preview at 1/2 stage
2	Preview of all stages on authentication

9. *checkDuplicate*

[+ V2.6.4] Check whether the scanned face is duplicated in the device.

10. *operationMode*

[+ V2.7.1] `FaceStation F2` Configures operation mode with below values, default is Fusion mode.

Value	Mode	Description	Default
0	Fusion Mode	Visual matching + IR matching	Default
1	Visual Mode	Visual matching	
2	Visual + IR	Visual matching, IR detects only face	

11. *maxRotation*

[+ V2.7.1] `FaceStation F2` When face is recognized normally it's front side.
Still, it is possible to determine how many degrees the image has been rotated from the front when FSF2 detects a face.

This enables detection failure in the case of images rotated over a certain angle.
maxRotation represents the maximum allowable value in this case, and the default value is 15 degrees.

12. *faceWidth*

[+ V2.7.1] FaceStation F2 This indicates the width of the face image, and you can specify the minimum and maximum values.
[+ 2.8.3] BioStation 3 The settings are ignored.

	Default(min)	Default(max)
FSF2	66	250
BS3	-	-

13. *searchRange*

[+ V2.7.1] FaceStation F2 Represents the face search range, and you can specify the x value (horizontal coordinate) of the range and the width from the x value point.
The default values of x and width are as follows.
[+ 2.8.3] BioStation 3 The settings are ignored.

14. *detectDistance* [+ 2.8.3] BioStation 3 This configures the

minimum and maximum detection range for facial recognition.. We no longer support faceWidth to pinpoint the face location using pixel units due to its complexity. Inst

ead
,
we
set
the
det
ecti
on
ran
ge
of
the
sub
ject
(fac
e).
The
unit
is
set
to
cm,
and
the
val
ue
mu
st
be
inp
utte
d
as
a
mul
tipl
e of
10.

15.
*wid
eSe
arc
h*
[+
2.8.
3]
Bio
Sta
tio

n 3
This
can
incr
eas
e
the
det
ecti
on
ran
ge
for
faci
al
rec
ogn
itio
n.
We
no
lon
ger
sup
port
sea
rch
Ran
ge
to
set
the
x-
coo
rdin
ate
and
wid
th
due
to
its
co
mpl
exit
y.
Inst
ead
,
we
set

the face detection setting as default (false), or a wide area(true). The details of the settings and protocols for the detection of wide area is set within the device, which the use

r can not cha nge .

If this sett ing is set to TRU E, the ca mer a det ect s sub ject s wit hin a larg e ran ge, and uni nte ntio nall y det ect and aut hen tica te mul tipl e sub ject

s at onc e. The refo re, the def ault sett ing is at fals e.

16. *unu sed* Res erv ed spa ce.

17. *res erv ed* Res erv ed spa ce.

BS 2R s4 85 Co nfi gE X

```
ty
pe
de
f
st
```

```
ruct{
uint32_t baudRate;
uint8_t channelIndex;
uint8_t useRegistance;
uint8_t numOfDevices;
uint8_t rese
```

```
rv
ed
[1
];
BS
2R
s4
85
Sl
av
eD
ev
ic
eE
X
sl
av
eD
ev
ic
es
[B
S2
_R
S4
85
_M
AX
_S
LA
VE
S_
PE
R_
CH
AN
NE
L]
;
}
BS
2R
s4
85
Ch
an
ne
lE
X;

ty
```

```
pe
de
f
st
ru
ct
{
ui
nt
8_
t
mo
de
[B
S2
_R
S4
85
_M
AX
_C
HA
NN
EL
S_
EX
];
ui
nt
8_
t
nu
mO
fC
ha
nn
el
s;
ui
nt
8_
t
re
se
rv
ed
[2
];
ui
nt
8_
```

```
t
re
se
rv
ed
1[
32
];
BS
2R
s4
85
Ch
an
ne
lE
X
ch
an
ne
ls
[B
S2
_R
S4
85
_M
AX
_C
HA
NN
EL
S]
;
}
BS
2R
s4
85
Co
nf
ig
EX
;
```

1.
bau
dRa
te
The
RS-

485 communication speed which can be configured as below.

Value
9600
19200
38400
57600
115200

2. *channelIndex* Communication channel index of the RS-485 network.

3. *useRegistance*

Decides whether to use a resistance.

4. *numOfDevices*
Number of slave devices.

5. *slaveDevices*
List of slave devices, which can be configured up to 32

dev
ices
.

6.
*mo
de*
Dec
ides
the
ope
rati
ng
mo
de
on
the
RS-
485
net
wor
k.

Value	Description
0	Not use
1	Master
2	Slave
3	Standalone

7.
*nu
mO
fCh
ann
els*
Nu
mb
er
of
RS-
485
cha
nne
l.

8.
*res
erv
ed*
Res
erv
ed
spa
ce.

9. *reserved1* Reserved space.

10. *channels* List of RS-485 channels, which can be configured up to 8 channels.

BS2CardConfigEx

```
typedef struct
```

```
{
uint8_t oid_ADF[13];
///
//valid value//{0x2A,0x85,0x70,0x81,0x1E,0x10,0x00,0x07,0x00,0x00,0x02,0x00,
```

```
0x00}
uint8_t size_ADF;
//uint8_t reserved1[2];
//uint8_t oid_DataObjectID[8];
uint16_t size_DataObjec
```

```
t[8];
uint8_t primaryKeyAuth[16];
//valid value
uint8_t secondaryKeyAuth[16];
///
//valid value
uint8_
```

```
t
re
se
rv
ed
2[
24
];
}
BS
2S
eo
sC
ar
d;
ty
pe
de
f
st
ru
ct
{
BS
2S
eo
sC
ar
d
se
os
;
ui
nt
8_
t
re
se
rv
ed
[2
4]
;
}
BS
2C
ar
dC
on
fi
gE
```

```
x;
```

1. *oid_ADF* ADF Address (Non changable)

2. *size_ADF* ADF size

3. *reserved1* Reserved sapce

4. *oid_DataObjectID* DataObject ID

5. *size_DataObject* DataOb

ject size

6. *primaryKeyAuth* Primary encryption key to access the Seos card information

7. *secondaryKeyAuth* Secondary encryption key to access the Seos card information

8. *reserved2* Reserved space

9. *seos* BS2SeosCard information

10. *reserved* Reserved space

BS2DstConfig

```
enum{
BS2_MAX_DST_SCHE
```

```
DU
LE
=
2,
};

ty
pe
de
f
st
ru
ct
{
ui
nt
16
_t
ye
ar
;
//
ye
ar
,
0
me
an
s
ev
er
y
ye
ar
.
ui
nt
8_
t
mo
nt
h;
//
[0
,
11
]
:
mo
nt
hs
```

```
since January int8_t ordinal; // [0, -1]: first, second, ..., last uint8_t weekDay; // [0, 6]: days since Sund
```

```
ay
uint8_t hour; // [0, 23]
uint8_t minute; // [0, 59]
uint8_t second; // [0, 59]
} BS2WeekTime;

typedef
```

```
struct {
BS2WeekTimeStartTime;
BS2WeekTimeEndTime;
int32_t timeOffset; // in seconds
uint8_t reserved[4];
```

```
}
BS
2D
st
Sc
he
du
le
;

ty
pe
de
f
st
ru
ct
{
ui
nt
8_
t
nu
mS
ch
ed
ul
es
;
ui
nt
8_
t
re
se
rv
ed
[3
1]
;

BS
2D
st
Sc
he
du
le
sc
he
du
```

```
les[
BS
2_
MA
X_
DS
T_
SC
HE
DU
LE
];
}
BS
2D
st
Co
nf
ig
;
```

1.
*yea
r*
Me
ans
yea
r,
and
if
set
to
0, it
me
ans
yea
rly.

2.
*mo
nth*
Me
ans
mo
nth,
and
has
a
val
ue

bet
wee
n 0
and
11
[Jan
uar
y-
Dec
em
ber]
.

3.
*ordi
nal*
It
star
ts
wit
h 0
and
me
ans
the
ord
er
of
first
,
sec
ond
,
etc.

4.
*wee
kDa
y*
Day
of
the
wee
k, 0
me
ans
Sun
day
, 1
me
ans

Monday.

5. *hour* Specifies the time in 24-hour format.

6. *minute* Specifies the minute.

7. *second* Specifies the seconds.

8. *startTime* It means start dat

e
and
tim
e.

9.
end
Tim
e
It
me
ans
the
end
dat
e.

10.
tim
eO
ffse
t
You
can
app
ly
the
DST
tim
e in
sec
ond
s.
For
exa
mpl
e, if
you
wa
nt
to
app
ly 1
hou
r,
ent
er
360
0.

11.

reserved Reserved space.

12. *numSchedules* The number of DST schedules to apply.

13. *schedules* DST schedule, up to two can be specified.

BS2Con

figs

```
typedef struct {
uint32_t config
Mask;
BS2Factory
Configfa
ctoryCo
nfig;
BS2S
ystemCon
fig
system
Config
;
```

```
BS
2A
ut
hC
on
fi
g
au
th
Co
nf
ig
;
BS
2S
ta
tu
sC
on
fi
g
st
at
us
Co
nf
ig
;
BS
2D
is
pl
ay
Co
nf
ig
di
sp
la
yC
on
fi
g;
BS
2I
pC
on
fi
g
ip
Co
```

```
nf
ig
;
BS
2I
pC
on
fi
gE
xt
ip
Co
nf
ig
Ex
t;
BS
2T
NA
Co
nf
ig
tn
aC
on
fi
g;
BS
2C
ar
dC
on
fi
g
ca
rd
Co
nf
ig
;
BS
2F
in
ge
rp
ri
nt
Co
nf
ig
fi
```

```
ng
er
pr
in
tC
on
fi
g;
BS
2R
s4
85
Co
nf
ig
rs
48
5C
on
fi
g;
BS
2W
ie
ga
nd
Co
nf
ig
wi
eg
an
dC
on
fi
g;
BS
2W
ie
ga
nd
De
vi
ce
Co
nf
ig
wi
eg
an
dD
```

```
ev
ic
eC
on
fi
g;
BS
2I
np
ut
Co
nf
ig
in
pu
tC
on
fi
g;
BS
2W
la
nC
on
fi
g
wl
an
Co
nf
ig
;
BS
2T
ri
gg
er
Ac
ti
on
Co
nf
ig
tr
ig
ge
rA
ct
io
nC
on
```

```
fi
g;
BS
2E
ve
nt
Co
nf
ig
ev
en
tC
on
fi
g;
BS
2W
ie
ga
nd
Mu
lt
iC
on
fi
g
wi
eg
an
dM
ul
ti
Co
nf
ig
;
BS
1C
ar
dC
on
fi
g
ca
rd
1x
Co
nf
ig
;
BS
```

```
2S
ys
te
mC
on
fi
gE
xt
sy
st
em
Ex
tC
on
fi
g;
BS
2V
oi
pC
on
fi
g
vo
ip
Co
nf
ig
;
BS
2F
ac
eC
on
fi
g
fa
ce
Co
nf
ig
;
}
BS
2C
on
fi
gs
;
```

1.

configMask Mask value of the configuration to be retrieved or set.

BS2IPV6Config

```
enum{
BS2_MAX_IPV6_ALLOC
```

```
AT
ED
_A
DD
R
=
8,
};

ty
pe
de
f
st
ru
ct
{
ui
nt
8_
t
us
eI
PV
6;
ui
nt
8_
t
re
se
rv
ed
1;
ui
nt
8_
t
us
eD
hc
pV
6;
ui
nt
8_
t
us
eD
ns
V6
```

```
;
uint8_t reserved[1];
char staticIpAddressV6[BS2_IPV6_ADDR_SIZE];
char staticGatewayV6[BS2_IPV6_ADDR_SIZE
```

```
];
ch
ar
dn
sA
dd
rV
6[
BS
2_
IP
V6
_A
DD
R_
SI
ZE
];
ch
ar
se
rv
er
Ip
Ad
dr
es
sV
6[
BS
2_
IP
V6
_A
DD
R_
SI
ZE
];
ui
nt
16
_t
se
rv
er
Po
rt
V6
;
ui
```

```
nt
16
_t
ss
lS
er
ve
rP
or
tV
6;
ui
nt
16
_t
po
rt
V6
;
ui
nt
8_
t
nu
mO
fA
ll
oc
at
ed
Ad
dr
es
sV
6;
ui
nt
8_
t
nu
mO
fA
ll
oc
at
ed
Ga
te
wa
yV
6;
```

```
uint8_t reserved[8];
char allocatedIpAddressV6[BS2_IPV6_ADDR_SIZE][BS2_MAX_IPV6_ALLOCATED_ADDR];
char allo
```

```
ca
te
dG
at
ew
ay
V6
[B
S2
_I
PV
6_
AD
DR
_S
IZ
E]
[B
S2
_M
AX
_I
PV
6_
AL
LO
CA
TE
D_
AD
DR
];
}
BS
2I
pC
on
fi
g;
```

1.
use IPV 6
Fla
g
indi
cati
ng
wh
eth

er to use IP V6.

2. *reserved1* Reserved space.

3. *useDhcpV6* Flag indicating whether to use DHCP.

4. *useDnsV6* Decides whether to use server address or server

URL
.

5.
stat
icIp
Add
res
sV6
Stat
ic IP
V6
add
res
s of
curr
ent
dev
ice.

6.
stat
icG
ate
way
V6
Stat
ic IP
V6
add
res
s of
gat
ewa
y.

7.
dns
Add
rV6
DN
S
add
res
s.

8.
ser
verI
pAd
dre

ssV6 IP address of BioStar. Used only in the server mode.

9. *serverPortV6* Port number of BioStar. Used only in the server mode.

10. *sslServerPortV6* Used wh

en
the
con
nec
tion
Mo
de
is
set
as
ser
ver
SSL
mo
de,
whi
ch
is
the
port
of
the
SDK
app
lica
tion
.

11.
*port
V6*
Port
nu
mb
er
of
the
dev
ice.

12.
*nu
mO
fAll
oca
ted
Add
res
sV6*
The
nu

mb
er
of
IP
V6
add
res
s
curr
entl
y
assi
gne
d to
the
dev
ice.

13.
*nu
mO
fAll
oca
ted
Gat
ewa
yV6*
The
nu
mb
er
of
gat
ewa
y
add
res
s
curr
entl
y
assi
gne
d to
the
dev
ice.

14.
*res
erv

ed	Reserved space.
15. *allocatedIpAddressV6*	The IPV6 address currently assigned to the device.
16. *allocatedGatewayV6*	The gateway address currently assigned to

the dev ice.

BS2DesFireCardConfigEx

```
typedef struct{
uint8_t appMasterKey[16];
uint8_t fileReadKey[16];
```

```
uint8_t fileWriteKey[16];
uint8_t fileReadKeyNumber;
uint8_t fileWriteKeyNumber;
uint8_t reserved[2];
}
```

```
BS
2D
es
Fi
re
Ap
pL
ev
el
Ke
y;
//
/<
52
by
te
s

ty
pe
de
f
st
ru
ct
{
BS
2D
es
Fi
re
Ap
pL
ev
el
Ke
y
de
sf
ir
eA
pp
Ke
y;
//
/<
52
by
te
s
ui
```

```
nt8_t reserved[16];
}BS2DesFireCardConfigEx;
//</68bytes
```

1. *appMasterKey* Application master key of DesFire.

2. *fileReadKey*

The key used to read the file.

3. *fileWriteKey* The key used to write the file.

4. *fileReadKeyNumber* The index of the key for reading the file.

5. *fileWriteKeyNumber* The

index of the key for writing the file.

6. *reserved* Reserved space.

7. *desfireAppKey* A structure containing DesFire key information.

8. *reserved* Reserved space.

BS2AuthConfigExt

```
typedef struct {
uint32_t extAuthScheduleBS2_MAX_NUM_OF_EXT_AUTH_MODE];
uint8_t useG
```

```
lo
ba
lA
PB
;
ui
nt
8_
t
gl
ob
al
AP
BF
ai
lA
ct
io
n;
ui
nt
8_
t
us
eG
ro
up
Ma
tc
hi
ng
;
ui
nt
8_
t
re
se
rv
ed
;

ui
nt
8_
t
re
se
rv
ed
2[
```

```
4]
;

uint8_t usePrivateAuth;
uint8_t faceDetectionLevel;
uint8_t useServerMatching;
uint8_t useFullAcc
```

```
es
s;

ui
nt
8_
t
ma
tc
hT
im
eo
ut
;
ui
nt
8_
t
au
th
Ti
me
ou
t;
ui
nt
8_
t
nu
mO
pe
ra
to
rs
;
ui
nt
8_
t
re
se
rv
ed
3[
1]
;

st
ru
ct
{
```

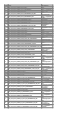
```
char userID[BS2_USER_ID_SIZE];
uint8_t level;
uint8_t reserved[3];
}
operators[BS2_MAX_OPERATORS];

uint
```

```
8_t reserved4[256];
}BS2AuthConfigExt;
```

1. *extAuthSchedule*
Schedule values to operate when each authentication mode is activated. It

has the following meanings for each value. If the value in the array is greater than 0, the authentication mode is activated. In the explanations below, biometric information me

ans
fing
erp
rint
s or
fac
es
dep
end
ing
on
the
dev
ice.



2.
*use
Glo
bal
APB*
This
flag
det
erm
ines
wh
eth
er
to
ena
ble
Glo
bal
APB
zon
e.

3.
*glo
bal
APB
Fail
Acti
on*
This
is a
basi
c

action to be performed when the device cannot query the server for Global APB violation.

Value	Description
0	Do not check APB
1	Soft APB
2	Hard APB

4. *useGroupMatching* Enables facial group matching.

5. *reserved* Res

erv
ed
spa
ce.

6.
*res
erv
ed2*
Res
erv
ed
spa
ce.

7.
*use
Priv
ate
Aut
h*
Ena
ble
priv
ate
aut
hen
tica
tion
mo
de.

8.
*fac
eDe
tect
ion
Lev
el*
This
is
the
fac
e
det
ecti
on
lev
el
val
ue

wh
en
aut
hen
tica
ting
the
use
r in
Bio
Stat
ion
A2,
and
if a
fac
e is
det
ect
ed
at a
lev
el
low
er
tha
n
the
spe
cifi
ed
lev
el,
it is
trea
ted
as
an
aut
hen
tica
tion
fail
ure.
Wh
en
ena
ble
d,
the
ca

mer
a
vie
w
acc
ordi
ng
to
Nor
mal
/Stri
ct is
dis
pla
yed
,
and
acc
ess
is
den
ied
if
the
ima
ge
log
is
not
rec
ogn
ized
as
a
fac
e
wh
en
suc
ces
sful
aut
hen
tica
tion
.
The
def
ault
is
0.

Value	Description
0	Do not detect face
1	Normal mode
2	Strict mode

It is available only in BioStation A2, not available in FaceStation 2 or FaceLite.

9. *useServerMatching* Enable server matching for fingerprint matching

or
faci
al
mat
chi
ng.

10.
*use
Full
Acc
ess*
This
par
am
eter
is
not
in
use
.

11.
*mat
chT
ime
out*
The
ma
xim
um
res
pon
se
tim
e in
fing
erp
rint
or
faci
al
mat
chi
ng
and
the
unit
is
sec
ond

s(se
c).

12.
*aut
hTi
me
out*
The
ma
xim
um
res
pon
se
tim
e in
use
r
aut
hen
tica
tion
and
the
unit
is
sec
ond
s(se
c).

13.
*nu
mO
per
ator
s*
The
nu
mb
er
of
ope
rato
rs.

14.
*res
erv
ed3*

Reserved

15. *userID* User ID

16. *level* It specifies the corresponding level of the user when the user is authenticated.

Value	Description
0	No level
1	Operator level
2	System configuration level
3	User information level

CAUTION You must specify

the number of operators to be added in the field ***numOperators*** when adding operators.	
17. *reserved*	Reserved
18. *reserved4*	Reserved

BS2FaceConfigExt

```
typedef struct{
uint8_t thermalCheckMode;
uint8_t maskCheckMode;
uint8_t reserved[2
```

```
];

uint8_t thermalFormat;
uint8_t reserved2;

uint16_t thermalThresholdLow;
uint16_t thermalThresholdHigh
```

```
;
uint8_t
maskDetectionLevel
;
uint8_t
auditTemperature
;

uint8_t
useRejectSound
;
uint8_t
useOverlapTh
```

```
er
ma
l;
ui
nt
8_
t
us
eD
yn
am
ic
RO
I;
ui
nt
8_
t
fa
ce
Ch
ec
kO
rd
er
;
}
BS
2F
ac
eC
on
fi
gE
xt
;
```

1.
ther
mal
Che
ckM
ode
Set
s
the
ther
mal
che
ck
mo

de.
Wh
en
set
to
HA
RD,
acc
ess
is
den
ied
if
exc
eed
ing
the
ther
mal
Thr
esh
old.
Wh
en
set
to
SOF
T,
acc
ess
is
not
affe
cte
d
eve
n if
exc
eed
ing
the
ther
mal
Thr
esh
old
but
lea
ves
a
rela

ted log. If thermal CheckMode is set to No use (0), The settings of thermal Format, thermal Threshold, auditTemperature, and use Overlap Thermal are ignored. And the reject sound due

to ther mal che ck by use Rej ect Sou nd, the tem per atur e che ck by fac eCh eck Ord er is ign ore d.

2. *ma skC hec kMo de* Fac eSt ati on F2 Set s the Mas k Che ck Mo de.

FaceStation 2
This setting is ignored. When set to HARD, access is denied if not detecting any mask on the face based on maskDetectionLevel. When set to SOFT, acc

ess is not affected even if not detecting any mask on the face based on maskDetectionLevel but leaves a related log. If maskCheckMode is set to No use (0), The setting of

maskDetectionLevel is ignored. And the reject sound due to mask detection check by useRejectSound, the mask detection check by faceCheckOrder is ignored.

3.
res erv ed
Res erv ed

4.
ther mal For mat
Rep res ent s the tem per atur e unit . You ma y cho ose the unit in Fah ren heit or Cel sius

5.
res erv ed2
Res erv ed

6.
ther mal

Threshold Low Supported version: FaceStation F2 V1.0.2, FaceStation 2 V1.5.0 This is the range value for determining high temperature and must be entered as

a val ue mul tipli ed by 100 of the tem per atur e to be set. Als o, you can onl y ent er in deg ree s Cel sius . This val ue is the basi s for the den ial of aut hen tica tion , and the sett

ing ran ge is bet wee n 100 (1º) and 450 0 (45 º). The def ault val ue is 320 0 (32 º), and if a val ue larg er or sm alle r tha n the sett ing ran ge is ent ere d, the def ault val ue a

set to 3200 (32º). And you must set a value less than thermal Threshold High.

7. *thermal Threshold High* This is the range value for determining high temperature

and must be entered as a value multiplied by 100 of the temperature to be set. Also, you can only enter in degrees Celsius. This value is the basis for the denial of

authentication, and the setting range is between 100 (1º) and 4500 (45º). The default value is 3800 (38º), and if a value larger or smaller than the setting range is ent

ere
d,
the
def
ault
val
ue
a
set
to
380
0
(38
º).
And
you
mu
st
set
a
val
ue
gre
ater
tha
n
ther
mal
Thr
esh
old
Low
.

8.
*ma
skD
ete
ctio
nLe
vel*
Fac
eSt
ati
on
F2
Set
s
the
ma
sk

detection level. The detecting level is based on internal setting value. FaceStation 2 This setting is ignored.

9. *auditTemperature* Decides whether the measured tem

per atur e is rec ord ed in the log or not.

10. *use Rej ect Sou nd* Dec ides wh eth er it sou nds wh en reje ctin g a use r due to ther mal Thr esh old or ma skD ete ctio nLe vel.

11. *use Ove rlap*

Thermal
Displays a thermal image overlaid on the screen.

12. *useDynamicROI*
When set to true, when measuring temperature, the user's forehead is found and measu

red, not a fixed area.

13. *faceCheckOrder*
It defines the sequence of thermal check and mask detection and authentication. Because the user should touch the device

in the case of ID combination authentication or PIN combination authentication, it is important to decide whether the device authentication mode or afterwa

rd
esp
ecia
lly
in a
hig
h-
risk
env
iron
me
nt.

Value	Description	Default

Aut
he
nti
cati
on
bef
ore
Te
mp
era
tur
e
che
ck
or
Ma
sk
det
ect
ion
che
ck

Check after authentication	Default
Check before authentication	
Authentication check only	

BS2ThermalCameraC

on fig

```
typedef struct {
uint8_t distance;
uint8_t emissionRate;

struct {
uint16_t x;
uint16_t y;
uint16_t wi
```

```
dt
h;
ui
nt
16
_t
he
ig
ht
;
}
ro
i;

ui
nt
8_
t
us
eB
od
yC
om
pe
ns
at
io
n;
in
t8
_t
co
mp
en
sa
ti
on
Te
mp
er
at
ur
e;
}
BS
2T
he
rm
al
Ca
me
```

```
raConfig;
```

1. *distance*
The distance measured by the thermal imaging camera. The unit is cm, and the default is 100.

2. *emissionRate*
The emissivity of the sub

ject refl ecti ng hea t. It is rec om me nde d to ent er wit hin the [95/ 97/ 98] ran ge. If the sub ject is a hu ma n, 98 is rec om me nde d.

3. *roi* ROI (Re gio n of inte rest) refe rs to the

regi
on
of
inte
rest
.
It
can
be
spe
cifi
ed
thro
ugh
coo
rdin
ate
s
(x,
y)
and
ran
ge
(wi
dth,
hei
ght)
val
ues
wh
en
me
asu
ring
tem
per
atur
e
on
the
fac
e.

4.
*use
Bod
yCo
mp
ens
atio
n*

It decides whether to use the compensate the body temperature.

5. *compensationTemperature* There may be a slight difference between the actual body temper

atur
e
and
the
bod
y
tem
per
atur
e
me
asu
rem
ent
usi
ng
the
ca
mer
a,
and
you
can
corr
ect
the
diff
ere
nce
by
sett
ing
a
val
ue
her
e.
It
mu
st
be
as
the
val
ue
mul
tipli
ed
by
10
of

the temperature to be set. The value is available -50 ~ +50

BS2BarcodeConfig

```
typedef struct {
uint8_t useBarcode;
uint8_
```

```
t
sc
an
Ti
me
ou
t;
ui
nt
8_
t
by
pa
ss
Da
ta
;
ui
nt
8_
t
tr
ea
tA
sC
SN
;

ui
nt
8_
t
re
se
rv
ed
[1
2]
;
}
BS
2B
ar
co
de
Co
nf
ig
;
```

1.

use Barcode Barcode usage flag.

2. *scanTimeout* Set the Barcode scan time. The unit is in seconds. The default is 4 seconds, and can be entered within a range of 4

to 10 seconds.

3. *bypassData* [+2.8.2] Used to send read barcode information to the server, not processed by the device. If the barcode value is stored in the use

r
info
rma
tion
stru
ctur
e
for
use
r
aut
hen
tica
tion
,
The
re
is a
size
con
stra
int
of
32
byt
es
(BS
2CS
NCa
rd
dat
a),
Call
the
BS2
_Se
tBa
rco
deS
can
List
ene
r,
use
this
opti
on
to
sen
d
bar

codes up to 512 bytes of size to the server.

4. *treatAsCSN* [+2.8.2] Indicates whether the Barcode should be treated the same as a regular CSN card. It is applied

from XS2-QR 1.1.3 and in the case of false, it is treated the same as before. This allows you to freely specify character sets that can be treated as barcodes from ASCII codes 32

to 126. (See description in BS2_WriteQRCode) If set to true, the barcode is treated like a number just like the existing CSN. Therefore, if you want to set the barcode

card data with special characters and English characters. In this case, only the card type may be different, and the CSN card and barcode data may be used in

the same value.

5. *reserved* Reserved space.

BS2InputConfigEx

```
typedef struct {
uint8_t numInputs;
uint8_t numSuper
```

```
vi
se
d;
ui
nt
8_
t
re
se
rv
ed
[1
8]
;

st
ru
ct
{
ui
nt
8_
t
po
rt
In
de
x;
ui
nt
8_
t
sw
it
ch
Ty
pe
;
ui
nt
16
_t
du
ra
ti
on
;

ui
nt
8_
```

```
t
re
se
rv
ed
;
ui
nt
8_
t
su
pe
rv
is
ed
Re
si
st
or
;
ui
nt
8_
t
re
se
rv
ed
1[
16
];

ui
nt
8_
t
re
se
rv
ed
2[
26
];
}
in
pu
ts
[B
S2
_M
AX
```

```
_INPUT_NUM_EX];

uint8_t reserved2[200];
}BS2InputConfigEx;
```

1. *numInputs* Number of Inputport.

2. *numSupervis*

ed
Nu
mb
er
of
sup
ervi
sed
inp
ut
port
.

3.
res
erv
ed
Res
erv
ed
Spa
ce.

4.
port
Ind
ex
Inp
ut
Port
Nu
mb
er.

5.
swit
chT
ype
Inp
ut
Sig
nal
Typ
e.

Value	Description
0	Normally Open
1	Normally Closed

6.
dur
atio
n

Input Signal Duration Time Measurement is milliseconds(ms).

7. *reserved* Reserved Space.

8. *supervised Resistor* You can set Supervised input resistance value typ

e or uns upe rvis e it.

9. *res erv ed1* Res erv ed Spa ce.

10. *res erv ed2* Res erv ed Spa ce.

11. *res erv ed2* Res erv ed Spa ce.

BS2RelayActionConfig

ty

```
pe
de
f
st
ru
ct
{
ui
nt
32
_t
de
vi
ce
ID
;
//
/<
4
by
te
s
ui
nt
8_
t
re
se
rv
ed
[1
6]
;
//
/<
16
by
te
s

st
ru
ct
{
ui
nt
8_
t
po
rt
;
```

```
//
/<
1
by
te
(r
el
ay
po
rt
)
ui
nt
8_
t
re
se
rv
ed
0;
//
/<
1
by
te
ui
nt
8_
t
di
sc
on
nE
na
bl
ed
;
//
/<
1
by
te
(R
S4
85
di
sc
on
ne
ct
io
```

```
n)
uint8_t reserved[9]; //</9 bytes

struct{
uint8_t port; //</1 byte(input port)
uint8_t type; //</1 by
```

```
te
(l
in
ka
ge
/l
at
ch
in
g/
re
le
as
e)
ui
nt
8_
t
ma
sk
;
//
/<
1
by
te
(a
la
rm
/f
au
lt
)
ui
nt
8_
t
re
se
rv
ed
[9
];
//
/<
9
by
te
s
}
in
```

```
put[BS2_MAX_RELAY_ACTION_INPUT];
///< 192 bytes
}
relay[BS2_MAX_RELAY_ACTION];
///< 816 bytes

uint8_t re
```

```
se
rv
ed
2[
15
2]
;
//
/<
15
2
by
te
s
}
BS
2R
el
ay
Ac
ti
on
Co
nf
ig
;
```

1. *deviceID* DeviceIdentifier

2. *reserved* ReservedSpace.

3. *relay* Rel

ay Setting Information

4. *port* Relay port Number.

5. *reserved0* Reserved Space.

6. *disconnEnabled* If set to true, a signal is made when RS485 is disconnected.

7. *reserved* Reserved Space.

8. *input* Defines to which input ports the relay ports will take action.

9. *port* Input port Identifier.

10. *type* Defines in which inp

ut typ e the inp ut will tak e acti on. If set to Link age , sig nal can be ma de wh en alar m is set to ma sk.

11. *ma sk* Set ma sk to Inp ut Sig nal Info .

12. *reserved* Reserved Space.

13. *reserved2* Reserved Space.

BS2VoipConfigExt

```
typedef struct {
BS2_USER_ID phoneNumber;
```

```
char description[48*3];

uint8_t reserved[32];
}
BS2ExtensionNumber;

typedef struct {
BS2_BOOL enable
```

```
d;
BS
2_
BO
OL
us
eO
ut
bo
un
dP
ro
xy
;
ui
nt
16
_t
re
gi
st
ra
ti
on
Du
ra
ti
on
;

BS
2_
UR
L
ad
dr
es
s;
BS
2_
PO
RT
po
rt
;

st
ru
ct
{
ui
```

```
int8_t speaker; // 0 ~ 100
uint8_t mic; // 0 ~ 100
} volume; ///< 2 bytes

BS2_USER_ID id;
BS2_USER_ID pass
```

```
word;
BS2_USER_ID authorizationCode;

struct{
BS2_URL address;
BS2_PORT port;
uint8_t reserved[2];
}
outb
```

```
ou
nd
Pr
ox
y;

ui
nt
8_
t
ex
it
Bu
tt
on
;
//
/
*,
#,
0~
9
ui
nt
8_
t
re
se
rv
ed
1;
ui
nt
8_
t
nu
mP
ho
ne
Bo
ok
;
BS
2_
BO
OL
sh
ow
Ex
te
ns
```

```
ionNumber;

BS2ExtensionNumberphoneebook[128];

uint8_t reserved2[32];//</32bytes(reserved)}
BS2VoipC
```

```
onfigExt;
```

1. *phoneNumber* This is the extension.

2. *description* Display information.

3. *reserved* Reserved space.

4. *enabled* Sets whether the VoI

P
ext
ensi
on
feat
ure
is
ena
ble
d.

5.
*use
Out
bou
ndP
rox
y*
Set
s
wh
eth
er
the
Out
bou
nd
Pro
xy
Ser
ver
is
con
figu
red.

6.
*regi
stra
tion
Dur
atio
n*
The
cycl
e of
upd
atin
g
the
rele

van
t
info
rma
tion
to
the
SIP
ser
ver.
Set
in
sec
ond
s
and
mu
st
be
bet
wee
n
60
and
600
.

7.
*add
res
s*
Ent
er
the
IP
add
res
s of
the
SIP
ser
ver
(us
uall
y
Bio
Star
).

8.
port

Enter the SIP server port. The default port is 5060.

9. *speaker*
Enter the speaker volume information for the intercom in the range 0 to 100. The default value is 50.

10. *mic* Enter the microphone volume information for the intercom in the range 0 to 100. The default value is 50.

11. *id* Enter the ID to connect to the SIP server.

12. *pas*

swo rd
Spe cifi es the pas swo rd to con nec t to the SIP ser ver.

13. *aut hori zati onC ode*
The aut hen tica tion cod e val ue req uire d to con nec t to the SIP ser ver.

14. *out bou ndP rox y*
Ent

er Outbound proxy server information.

15. *address* Enter the IP address of the Outbound Proxy Server.

16. *port* Enter the Outbound Proxy Server port.

17. *reserv*

ed
Res
erv
ed
spa
ce.

18.
*exit
But
ton*
But
ton
sy
mb
ol
to
be
use
d
as
a
che
ck-
out
butt
on.

Value	Description
*	'*' ASCII code 42
#	'#' ASCII code 35
0~9	'0'~'9' ASCII code 48~57

19.
*res
erv
ed1*
Res
erv
ed
spa
ce.

20.
*nu
mP
hon
eBo
ok*
Nu
mb
er
of
pho

ne
boo
ks.

21.
*sho
wEx
ten
sion
Nu
mb
er*
Det
erm
ines
wh
eth
er
to
sho
w
the
pho
ne
boo
k.

22.
*pho
neb
ook*
You
can
spe
cify
up
to
128
ext
ensi
ons
in
you
r
pho
ne
boo
k.

23.
res

erv ed2 Res erv ed spa ce.

BS2RtspConfig

```
typedef struct {
BS2_USER_ID id;
BS2_USER_ID password;

BS2_URL addr
```

```
es
s;

BS
2_
PO
RT
po
rt
;
BS
2_
BO
OL
en
ab
le
d;
ui
nt
8_
t
re
se
rv
ed
;

ui
nt
8_
t
re
se
rv
ed
2[
32
];
}
BS
2R
ts
pC
on
fi
g;
```

1.
id
Acc

oun
t
info
rma
tion
wh
en
con
nec
ting
to
the
RTS
P
ser
ver.

2.
*pas
swo
rd*
Pas
swo
rd
wh
en
con
nec
ting
to
the
RTS
P
ser
ver.

3.
*add
res
s*
Ent
er
the
add
res
s of
the
RTS
P
ser
ver.

4. *port* Enter the RTSP server connection port. The default port is 554.

5. *enabled* Sets whether an RTSP connection is enabled.

6. *reserved* Reserved space.

7.
res
erv
ed2
Res
erv
ed
spa
ce.

1)

Maximum
Transmission Unit
2)

Most Significant Bit
3)

Least Significant Bit