Table of Contents

Device API

API that controls the device information or upgrades the firmware.

- BS2_GetDeviceInfo: Gets the device information.
- BS2_GetDeviceInfoEx: [+ 2.6.0] Gets additional device information.
- BS2_GetDeviceTime: Gets the device time.
- BS2_SetDeviceTime: Sets the device time.
- BS2_ClearDatabase: Initializes the user information and blacklist.
- BS2_FactoryReset: Initializes all configurations and the database.
- BS2_RebootDevice: Restarts the device.
- BS2_LockDevice: Doesn't allow user authentication by locking the device.
- BS2_UnlockDevice: Allows user authentication by unlocking the device.
- BS2_SetKeepAliveTimeout: Configures the keep-alive time of the device.
- BS2_UpgradeFirmware: Upgrades the firmware.
- BS2_UpdateResource: Updates the resource.
- BS2_GetSpecifiedDeviceInfo: [+ 2.6.3] Gets specified device information.
- BS2_GetAuthOperatorLevelEx: [+ 2.6.3] Gets specified device operator. (Support operator up to 1000)
- BS2_GetAllAuthOperatorLevelEx: [+ 2.6.3] Gets all device operators. (Support operator up to 1000)
- BS2_SetAuthOperatorLevelEx: [+ 2.6.3] Sets device operator. (Support operator up to 1000)
- BS2_RemoveAuthOperatorLevelEx: [+ 2.6.3] Removes specified device operator. (Support operator up to 1000)
- BS2_RemoveAllAuthOperatorLevelEx: [+ 2.6.3] Removes all device operators. (Support operator up to 1000)
- BS2_GetDeviceCapabilities: [+ 2.8] Gets available function information of the device.

Structure

BS2SimpleDeviceInfo

```
typedef struct
{
    uint32_t id;
    uint16_t type;
    uint8_t connectionMode;
    uint32_t ipv4Address;
    uint16_t port;
    uint32_t maxNumOfUser;
    uint8_t userNameSupported;
    uint8_t userPhotoSupported;
    uint8_t pinSupported;
    uint8_t cardSupported;
    uint8_t fingerSupported;
    uint8_t faceSupported;
    uint8_t wlanSupported;
```

```
    uint8_t tnaSupported;
    uint8_t triggerActionSupported;
    uint8_t wiegandSupported;
    uint8_t imageLogSupported;
    uint8_t dnsSupported;
    uint8_t jobCodeSupported;
    uint8_t wiegandMultiSupported;
    uint8_t rs485Mode;
    uint8_t sslSupported;
    uint8_t rootCertExist;
    uint8_t dualIDSupported;
    uint8_t useAlphanumericID;
    uint32_t connectedIP;
    uint8_t phraseCodeSupported;
    uint8_t card1xSupported;
    uint8_t systemExtSupported;
    uint8_t voipSupported;
}BS2SimpleDeviceInfo;
```

1. *id*
The device identifier which is always above 1.

2. *type*
Code value of device type.

Value	Description
0x01	BioEntry Plus
0x02	BioEntry W
0x03	BioLite Net
0x04	Xpass
0x05	Xpass S2
0x06	Secure IO 2
0x07	DM-20
0x08	BioStation 2
0x09	BioStation A2
0x0A	FaceStation 2
0x0B	IO Device
0x0C	BioStation L2
0x0D	BioEntry W2
0x0E	CoreStation
0x0F	Output Module
0x10	Input Module
0x11	BioEntry P2
0x0F	OM-120

3. *connectionMode*
It indicates the connection mode between the BioStar application and device which is separated by the subject of the connection as direct mode(0x0) and server mode(0x1). The BioStar application connects to the device in direct mode, and the device connects to the BioStar application in server

mode. The default settings for the devices are direct mode, and to change the connection mode refer to IP Config.

4. *ipv4Address*
IP address of the selected device.

5. *port*
TCP port number of the selected device.

6. *maxNumOfUser*
Maximum capacity of users that can be stored in the device.

7. *userNameSupported*
Flag that notifies whether the device supports user name.

8. *userPhotoSupported*
Flag that notifies whether the device supports user profile picture.

9. *pinSupported*
Flag that notifies whether the device supports PIN.

10. *cardSupported*
Flag that notifies whether the device supports Smart card authentication.

11. *fingerSupported*
Flag that notifies whether the device supports finger authentication.

12. *faceSupported*
Flag that notifies whether the device supports face recognition.

13. *wlanSupported*
Flag that notifies whether the device supports wireless LAN.

14. *tnaSupported*
Flag that notifies whether the device supports time and attendance.

15. *triggerActionSupported*
Flag that notifies whether the device supports trigger action.

16. *wiegandSupported*
Flag that notifies whether the device supports wiegand.

17. *imageLogSupported*
Flag that notifies whether the device supports image logs.

18. *dnsSupported*
Flag that notifies whether the device supports DNS.

19. *jobCodeSupported*
Flag that notifies whether the device supports job codes.

20. *wiegandMultiSupported*
Flag that notifies whether the device supports Multi-Wiegand.

21. *rs485Mode*
RS-485 mode of the device.

22. *sslSupported*
Flag that notifies whether the device supports SSL communication.

23. *rootCertExist*
Flag that notifies whether the device has a root certificate.

24. *dualIDSupported*
Flag that notifies whether the device supports alphanumeric ID.

25. *useAlphanumericID*
Flag that notifies whether the device is currently using Alphanumeric ID.

26. *connectedIP*
IP address where the device is connected to. (0xFFFFFFFF if disconnected)

27. *phraseCodeSupported*
Flag that notifies whether the device supports personal messages.

28. *card1xSupported*
Flag that notifies whether the device supports reading 1.x ToC cards.

29. *systemExtSupported*
Flag that notifies whether the device supports configuring RS-485 keys.

30. *voipSupported*
Flag that notifies whether the device supports VoIP.

BS2SimpleDeviceInfoEx

Retrieves BS2SimpleDeviceInfo and supported information.

```
typedef struct
{
    enum
    {
        BS2_SUPPORT_RS485EX     = 0x00000001,
        BS2_SUPPORT_CARDEX      = 0x00000002,
        BS2_SUPPORT_DST         = 0x00000004,
        BS2_SUPPORT_DESFIREEX   = 0x00000008,
        BS2_SUPPORT_FACE_EX = 0x00000010,

        BS2_SUPPORT_FINGER_SCAN = 0x00010000,
        BS2_SUPPORT_FACE_SCAN   = 0x00020000,
        BS2_SUPPORT_FACE_EX_SCAN = 0x00040000,

        BS2_SUPPORT_ALL         = BS2_SUPPORT_RS485EX | BS2_SUPPORT_CARDEX |
BS2_SUPPORT_DST | BS2_SUPPORT_DESFIREEX | BS2_SUPPORT_FACE_EX |
BS2_SUPPORT_FINGER_SCAN | BS2_SUPPORT_FACE_SCAN | BS2_SUPPORT_FACE_EX_SCAN,
```

```
    };
    uint32_t supported;
    uint8_t  reserved[4];
}BS2SimpleDeviceInfoEx;
```

1. *supported*
The current device additionally obtains information beyond the functionality provided by BS2SimpleDeviceInfo.
By bit masking with the values defined below, you can check if it is supported.

Definition	Value	Description
BS2_SUPPORT_RS485EX	0x00000001	Whether RS485 extensions are supported (In case of CoreStation 40)
BS2_SUPPORT_CARDEX	0x00000002	Whether iClass SEOS card is used
BS2_SUPPORT_DST	0x00000004	Whether daylight savings time is used
BS2_SUPPORT_DESFIREEX	0x00000008	Whether DesFire advanced setting is supported [+2.6.4]
BS2_SUPPORT_FACE_EX	0x00000010	Whether support face matching for FSF2 [+ V2.7.1]
BS2_SUPPORT_FINGER_SCAN	0x00010000	Whether support fingerprint scan [+ V2.7.1]
BS2_SUPPORT_FACE_SCAN	0x00020000	Whether support face scan for FS2 and FL [+ V2.7.1]
BS2_SUPPORT_FACE_EX_SCAN	0x00040000	Whether support face scan for FSF2 [+ V2.7.1]
BS2_SUPPORT_ALL	0x0000000F	Whether to provide additional full information

2. *reserved*
Reserved space.

BS2ResourceElement

```
typedef struct
{
    uint8_t type;
    uint32_t numResData;
    struct {
        uint8_t index;
        uint32_t dataLen;
        uint8_t* data;
    }resData[128];
}BS2ResourceElement;
```

1. *type*
Resource data type.

Value	Description	Supported data format
0	UI(Langauge pack)	Suprema language pack
1	Notice message	UTF-8 string
2	Image(Background)	PNG
3	Slide image	PNG
4	Sound	WAVE

2. *numResData*
Number of resource data.

3. *index*
Resource index number.

4. *dataLen*
Resource data length.

5. *data*
Binary resource data.

BS2IPv6DeviceInfo

```
enum {
    BS2_MAX_IPV6_ALLOCATED_ADDR = 8,
};

typedef struct
{
    BS2_DEVICE_ID id;
    uint8_t reserved[1];
    uint8_t bIPv6Mode;
    char ipv6Address[BS2_IPV6_ADDR_SIZE];
    uint16_t portV6;
    char connectedIPV6[BS2_IPV6_ADDR_SIZE];
    uint8_t numOfAllocatedAddressV6;
    char
allocatedIpAddressV6[BS2_IPV6_ADDR_SIZE][BS2_MAX_IPV6_ALLOCATED_ADDR];
}BS2IPv6DeviceInfo;
```

1. *id*
Device ID

2. *reserved*
Reserved space

3. *bIPv6Mode*
Flag to determine whether to work IPv6 mode or not.

4. *ipv6Address*

IPv6 address of device

5. *portV6*
IPv6 port of device

6. *connectedIPV6*
IPv6 address of server which device is connected.

7. *numOfAllocatedAddressV6*
Number of IPv6 addresses currently allocated to device. 8. *allocatedIpAddressV6*
IPv6 addresses currently allocated to device.

BS2AuthOperatorLevel

```
typedef struct {
    char userID[BS2_USER_ID_SIZE];
    uint8_t level;
    uint8_t reserved[3];
} BS2Operator;

typedef BS2Operator BS2AuthOperatorLevel;
```

1. *userID*
User ID

2. *level*
Sets operator level when user authenticates.

Value	Description
0	No auth
1	Administrator level
2	System configuration level
3	User information lelve

3. *reserved*
Reserved space

BS2DeviceCapabilities

[+ 2.8]

```
typedef struct {
    uint32_t maxUsers;                ///< 4 bytes
    uint32_t maxEventLogs;            ///< 4 bytes
    uint32_t maxImageLogs;            ///< 4 bytes
    uint32_t maxBlacklists;           ///< 4 bytes
    uint32_t maxOperators;            ///< 4 bytes
    uint32_t maxCards;                ///< 4 bytes
```

```
    uint32_t maxFaces;                  ///< 4 bytes
    uint32_t maxFingerprints;           ///< 4 bytes
    uint32_t maxUserNames;              ///< 4 bytes
    uint32_t maxUserImages;             ///< 4 bytes
    uint32_t maxUserJobs;               ///< 4 bytes
    uint32_t maxUserPhrases;            ///< 4 bytes
    uint8_t maxOutputPorts;             ///< 1 byte
    uint8_t maxRelays;                  ///< 1 byte
    uint8_t maxRS485Channels;           ///< 1 byte

    uint8_t cameraSupported: 1;
    uint8_t tamperSupported: 1;
    uint8_t wlanSupported: 1;
    uint8_t displaySupported: 1;
    uint8_t thermalSupported: 1;
    uint8_t maskSupported: 1;
    uint8_t faceExSupported: 1;
    uint8_t unused: 1;

    union {
        uint32_t mask;                  ///< 4 bytes
        struct {
            uint32_t EM: 1;
            uint32_t HIDProx: 1;
            uint32_t MifareFelica: 1;
            uint32_t iClass: 1;
            uint32_t ClassicPlus: 1;
            uint32_t DesFireEV1: 1;
            uint32_t SRSE: 1;
            uint32_t SEOS: 1;
            uint32_t NFC: 1;
            uint32_t BLE: 1;
            uint32_t reserved: 21;
            uint32_t useCardOperation: 1;
        };
    } cardSupported;

    struct {
        BS2_BOOL extendedMode;          ///< 1 byte
        union {
            uint8_t mask;               ///< 1 byte
            struct {
                uint8_t card: 1;
                uint8_t fingerprint: 1;
                uint8_t face: 1;
                uint8_t id: 1;
                uint8_t pin: 1;
                uint8_t reserved: 3;
            };
        } credentials;
        uint8_t reserved[2];            ///< 2 bytes
```

```
        union {
            struct {
                union {
                    uint8_t mask;     ///< 1 byte
                    struct {
                        uint8_t biometricOnly: 1;
                        uint8_t biometricPIN: 1;
                        uint8_t unused: 6;
                    };
                } biometricAuth;

                union {
                    uint8_t mask;     ///< 1 byte
                    struct {
                        uint8_t cardOnly: 1;
                        uint8_t cardBiometric: 1;
                        uint8_t cardPIN: 1;
                        uint8_t cardBiometricOrPIN: 1;
                        uint8_t cardBiometricPIN: 1;
                        uint8_t unused: 3;
                    };
                } cardAuth;

                union {
                    uint8_t mask;     ///< 1 byte
                    struct {
                        uint8_t idBiometric: 1;
                        uint8_t idPIN: 1;
                        uint8_t idBiometricOrPIN: 1;
                        uint8_t idBiometricPIN: 1;
                        uint8_t unused: 4;
                    };
                } idAuth;
            } legacy;

            struct {
                union {
                    uint32_t mask;    ///< 4 bytes
                    struct {
                        uint32_t faceOnly: 1;
                        uint32_t faceFingerprint: 1;
                        uint32_t facePIN: 1;
                        uint32_t faceFingerprintOrPIN: 1;
                        uint32_t faceFingerprintPIN: 1;
                        uint32_t unused: 27;
                    };
                } faceAuth;

                union {
                    uint32_t mask;    ///< 4 bytes
                    struct {
```

```
            uint32_t fingerprintOnly: 1;
            uint32_t fingerprintFace: 1;
            uint32_t fingerprintPIN: 1;
            uint32_t fingerprintFaceOrPIN: 1;
            uint32_t fingerprintFacePIN: 1;
            uint32_t unused: 27;
        };
    } fingerprintAuth;

    union {
        uint32_t mask;    ///< 4 bytes
        struct {
            uint32_t cardOnly: 1;
            uint32_t cardFace: 1;
            uint32_t cardFingerprint: 1;
            uint32_t cardPIN: 1;
            uint32_t cardFaceOrFingerprint: 1;
            uint32_t cardFaceOrPIN: 1;
            uint32_t cardFingerprintOrPIN: 1;
            uint32_t cardFaceOrFingerprintOrPIN: 1;
            uint32_t cardFaceFingerprint: 1;
            uint32_t cardFacePIN: 1;
            uint32_t cardFingerprintFace: 1;
            uint32_t cardFingerprintPIN: 1;
            uint32_t cardFaceOrFingerprintPIN: 1;
            uint32_t cardFaceFingerprintOrPIN: 1;
            uint32_t cardFingerprintFaceOrPIN: 1;
            uint32_t unused: 17;
        };
    } cardAuth;

    union {
        uint32_t mask;    ///< 4 bytes
        struct {
            uint32_t idFace: 1;
            uint32_t idFingerprint: 1;
            uint32_t idPIN: 1;
            uint32_t idFaceOrFingerprint: 1;
            uint32_t idFaceOrPIN: 1;
            uint32_t idFingerprintOrPIN: 1;
            uint32_t idFaceOrFingerprintOrPIN: 1;
            uint32_t idFaceFingerprint: 1;
            uint32_t idFacePIN: 1;
            uint32_t idFingerprintFace: 1;
            uint32_t idFingerprintPIN: 1;
            uint32_t idFaceOrFingerprintPIN: 1;
            uint32_t idFaceFingerprintOrPIN: 1;
            uint32_t idFingerprintFaceOrPIN: 1;
            uint32_t unused: 18;
        };
    } idAuth;
```

```
            } extended;
        };
    } authSupported;

    uint8_t intelligentPDSupported: 1;
    uint8_t unused2: 7;

    uint8_t reserved[431];
} BS2DeviceCapabilities;
```

1. *maxUsers*
장치에 저장 가능한 정보의 최대 갯수를 나타냅니다. (사용자)

2. *maxEventLogs*
장치에 저장 가능한 정보의 최대 갯수를 나타냅니다. (이벤트로그)

3. *maxImageLogs*
장치에 저장 가능한 정보의 최대 갯수를 나타냅니다. (이미지로그)

4. *maxBlacklists*
장치에 저장 가능한 정보의 최대 갯수를 나타냅니다. (블랙리스트)

5. *maxOperators*
장치에 저장 가능한 정보의 최대 갯수를 나타냅니다. (관리자)

6. *maxCards*
장치에 저장 가능한 정보의 최대 갯수를 나타냅니다. (카드)

7. *maxFaces*
장치에 저장 가능한 정보의 최대 갯수를 나타냅니다. (얼굴)

8. *maxFingerprints*
장치에 저장 가능한 정보의 최대 갯수를 나타냅니다. (지문)

9. *maxUserNames*
장치에 저장 가능한 정보의 최대 갯수를 나타냅니다. (사용자 명)

10. *maxUserImages*
장치에 저장 가능한 정보의 최대 갯수를 나타냅니다. (사용자 이미지)

11. *maxUserJobs*
장치에 저장 가능한 정보의 최대 갯수를 나타냅니다. (Job code)

12. *maxUserPhrases*
장치에 저장 가능한 정보의 최대 갯수를 나타냅니다. (사용자 구문)

13. *maxCardsPerUser*
장치에 저장 가능한 정보의 최대 갯수를 나타냅니다. (사용자별 카드)

14. *maxFacesPerUser*

장치에 저장 가능한 정보의 최대 갯수를 나타냅니다. (사용자별 얼굴)

15. *maxFingerprintsPerUser*
장치에 저장 가능한 정보의 최대 갯수를 나타냅니다. (사용자별 지문)

16. *maxInputPorts*
장치에 저장 가능한 정보의 최대 갯수를 나타냅니다. (장치의 입력포트)

17. *maxOutputPorts*
장치에 저장 가능한 정보의 최대 갯수를 나타냅니다. (장치의 출력포트)

18. *maxRelays*
장치에 저장 가능한 정보의 최대 갯수를 나타냅니다. (장치의 릴레이)

19. *maxRS485Channels*
장치에 저장 가능한 정보의 최대 갯수를 나타냅니다. (RS485 채널)

20. *시스템 지원 정보*
장치가 지원 가능한 시스템 정보를 bit 단위로 아래와 같이 나타냅니다.

비트위치	비트 수	멤버명	설명
0	1	cameraSupported	카메라 지원 여부.
1	1	tamperSupported	탬퍼 지원 여부.
2	1	wlanSupported	무선랜 지원 여부.
3	1	displaySupported	화면의 지원 여부.
4	1	thermalSupported	열화상 카메라 지원 여부.
5	1	maskSupported	마스크 검출 지원 여부.
6	1	faceExSupported	Visual camera 인증 지원 여부.
7	1	unused	미할당 필드.

21. *cardSupported*
카드관련 지원 정보를 나타냅니다. mask값으로 전체를 또는 bit 단위로 각각의 지원항목에 접근할 수 있습니다.

비트위치	비트 수	멤버명	설명
-	전체	mask	전체정보
0	1	EM	EM 카드
1	1	HIDProx	HID Proximity 카드
2	1	MifareFelica	MIFARE / FeliCa
3	1	iClass	iClass 카드
4	1	ClassicPlus	Classic plus 카드
5	1	DesFireEV1	DESFire EV1
6	1	SRSE	iClass SR, iClass SE
7	1	SEOS	iClass SEOS
8	1	NFC	NFC 카드
9	1	BLE	BLE
10	21	reserved	미할당 필드.
31	1	useCardOperation	카드 사용 여부

22. *authSupported*
인증관련 지원 정보를 나타냅니다.

23. *extendedMode*
true인 경우, 확장인증모드를 지원하며, authSupported.extended를 참조합니다.
false인 경우 비확장인증 모드를 지원하며, authSupported.lagacy를 참조합니다.

24. *credentials*
지원되는 인증 수단을 나타냅니다. mask값으로 전체를 또는 bit 단위로 각각의 지원항목에 접근할 수 있습니다.

비트위치	비트 수	멤버명	설명
-	전체	mask	전체정보
0	1	card	카드
1	1	fingerprint	지문
2	1	face	얼굴
3	1	id	ID
4	1	pin	PIN
5	3	reserved	미할당 필드.

25. *reserved*
예약된 공간입니다.

26. *legacy*
비확장 인증모드 지원 시, 참조되는 정보입니다.

27. *biometricAuth*
(비확장 인증모드)Biometric 인증 조합을 나타냅니다.

비트위치	비트 수	멤버명	설명
-	전체	mask	전체정보
0	1	biometricOnly	Biometric only
1	1	biometricPIN	Biometric + PIN
2	6	unused	미할당 필드.

28. *cardAuth*
(비확장 인증모드)Card 인증 조합을 나타냅니다.

비트위치	비트 수	멤버명	설명
-	전체	mask	전체정보
0	1	cardOnly	Card only
1	1	cardBiometric	Card + Biometric
2	1	cardPIN	Card + PIN
3	1	cardBiometricOrPIN	Card + Biometric/PIN
4	1	cardBiometricPIN	Card + Biometric + PIN
5	3	unused	미할당 필드.

29. *idAuth*
(비확장 인증모드)ID 인증 조합을 나타냅니다.

비트위치	비트 수	멤버명	설명
-	전체	mask	전체정보
0	1	idBiometric	ID + Biometric
1	1	idPIN	ID + PIN
2	1	idBiometricOrPIN	ID + Biometric/PIN
3	1	idBiometricPIN	ID + Biometric + PIN
4	4	unused	미할당 필드.

30. *extended*
확장 인증모드 지원 시, 참조되는 정보입니다.

31. *faceAuth*
(확장 인증모드)얼굴 인증 조합을 나타냅니다.

비트위치	비트 수	멤버명	설명
-	전체	mask	전체정보
0	1	faceOnly	Face only
1	1	faceFingerprint	Face + Fingerprint
2	1	facePIN	Face + PIN
3	1	faceFingerprintOrPIN	Face + Fingerprint/PIN
4	1	faceFingerprintPIN	Face + Fingerprint + PIN
5	27	unused	미할당 필드.

32. *fingerprintAuth*
(확장 인증모드)지문 인증 조합을 나타냅니다.

비트위치	비트 수	멤버명	설명
-	전체	mask	전체정보
0	1	fingerprintOnly	Fingerprint only
1	1	fingerprintFace	Fingerprint + Face
2	1	fingerprintPIN	Fingerprint + PIN
3	1	fingerprintFaceOrPIN	Fingerprint + Face/PIN
4	1	fingerprintFacePIN	Fingerprint + Face + PIN
5	27	unused	미할당 필드.

33. *cardAuth*
(확장 인증모드)카드 인증 조합을 나타냅니다.

비트위치	비트 수	멤버명	설명
-	전체	mask	전체정보
0	1	cardOnly	Card only
1	1	cardFace	Card + Face
2	1	cardFingerprint	Card + Fingerprint
3	1	cardPIN	Card + PIN
4	1	cardFaceOrFingerprint	Card + Face/Fingerprint
5	1	cardFaceOrPIN	Card + Face/PIN
6	1	cardFingerprintOrPIN	Card + Fingerprint/PIN
7	1	cardFaceOrFingerprintOrPIN	Card + Face/Fingerprint/PIN

비트위치	비트 수	멤버명	설명
8	1	cardFaceFingerprint	Card + Face + Fingerprint
9	1	cardFacePIN	Card + Face + PIN
10	1	cardFingerprintFace	Card + Fingerprint + Face
11	1	cardFingerprintPIN	Card + Fingerprint + PIN
12	1	cardFaceOrFingerprintPIN	Card + Face/Fingerprint + PIN
13	1	cardFaceFingerprintOrPIN	Card + Face + Fingerprint/PIN
14	1	cardFingerprintFaceOrPIN	Card + Fingerprint + Face/PIN
15	17	unused	미할당 필드.

34. *idAuth*
(확장 인증모드)ID 인증 조합을 나타냅니다.

비트위치	비트 수	멤버명	설명
-	전체	mask	전체정보
1	1	idFace	ID + Face
2	1	idFingerprint	ID + Fingerprint
3	1	idPIN	ID + PIN
4	1	idFaceOrFingerprint	ID + Face/Fingerprint
5	1	idFaceOrPIN	ID + Face/PIN
6	1	idFingerprintOrPIN	ID + Fingerprint/PIN
7	1	idFaceOrFingerprintOrPIN	ID + Face/Fingerprint/PIN
8	1	idFaceFingerprint	ID + Face + Fingerprint
9	1	idFacePIN	ID + Face + PIN
10	1	idFingerprintFace	ID + Fingerprint + Face
11	1	idFingerprintPIN	ID + Fingerprint + PIN
12	1	idFaceOrFingerprintPIN	ID + Face/Fingerprint + PIN
13	1	idFaceFingerprintOrPIN	ID + Face + Fingerprint/PIN
14	1	idFingerprintFaceOrPIN	ID + Fingerprint + Face/PIN
15	18	unused	미할당 필드.

35. *시스템 지원 정보*
장치가 지원 가능한 시스템 정보를 bit 단위로 아래와 같이 나타냅니다.

비트위치	비트 수	멤버명	설명
0	1	intelligentPDSupported	Intelligent PD 지원 여부.
1	7	unused2	미할당 필드.

36. *reserved*
예약된 공간입니다.