Table of Contents

User Management API

API that provides functions to enroll and delete users.

- BS2_GetUserList: Gets the enrolled user ID list.
- BS2_RemoveUser: Deletes user.
- BS2_RemoveAllUser: Deletes all users.
- BS2_GetUserInfos: Gets the user information of the given user ID.
- BS2_GetUserInfosEx: [+ 2.4.0] Gets the user information of the given user ID. (including Job code and User phrase)
- BS2_EnrolUser: Enrolls new user.
- BS2_EnrolUserEx: [+ 2.4.0] Enrolls new user. (including Job code and User phrase)
- BS2_EnrollUser: [+ 2.6.3] Enrolls new user.
- BS2_EnrollUserEx: [+ 2.6.3] Enrolls new user. (including Job code and User phrase)
- BS2_GetUserDatas: Gets selected data of user.
- BS2_GetUserDatasEx: [+ 2.5.0] Gets selected data of user. (including Job code, User phrase)
- BS2_GetSupportedUserMask: Gets user settings supported by the device.
- BS2_EnrollUserSmall: [+ 2.6.3] Enrolls new user with efficient use of memory.
- BS2_EnrollUserSmallEx: [+ 2.6.3] Enrolls new user with efficient use of memory.
- BS2_GetUserSmallInfos: [+ 2.6.3] Gets the user information of the given user ID with efficient use of memory.
- BS2_GetUserSmallInfosEx: [+ 2.6.3] Gets the user information of the given user ID with efficient use of memory.
- BS2_GetUserSmallDatas: [+ 2.6.3] Gets selected data of user with efficient use of memory.
- BS2_GetUserSmallDatasEx: [+ 2.6.3] Gets selected data of user with efficient use of memory.
- BS2_EnrollUserFaceEx: [+ 2.7.1] `Visual Face Support` Enrolls new user.
- BS2_GetUserInfosFaceEx: [+ 2.7.1] `Visual Face Support` Gets the user information of the given user ID.
- BS2_GetUserDatasFaceEx: [+ 2.7.1] `Visual Face Support` Gets selected data of user.
- BS2_PartialUpdateUser: [+ 2.8.3] Updates partial information of an already registered user.
- BS2_PartialUpdateUserEx: [+ 2.8.3] Updates partial information of an already registered user. (including Job code and User phrase)
- BS2_PartialUpdateUserSmall: [+ 2.8.3] Updates partial information of an already registered user with efficient use of memory.
- BS2_PartialUpdateUserSmallEx: [+ 2.8.3] Updates partial information of an already registered user with efficient use of memory.
- BS2_PartialUpdateUserFaceEx: [+ 2.8.3] `Visual Face Support` Updates partial information of an already registered user.
- BS2_GetUserStatistic: [+ 2.8.3] Gets the user-related statistics that the device has.
- BS2_GetUserOverride: [+ 2.9.12] Gets user information for extended door open time that meets specified conditions.
- BS2_GetAllUserOverride: [+ 2.9.12] Gets all user information using extended door open time.
- BS2_SetUserOverride: [+ 2.9.12] Sets users to use extended door open time.
- BS2_RemoveUserOverride: [+ 2.9.12] Removes users using extended door open time.
- BS2_RemoveAllUserOverride: [+ 2.9.12] Removes all users using extended door open time.

Structure

BS2User

```
typedef struct {
    char userID[BS2_USER_ID_SIZE];
    uint8_t formatVersion;
    uint8_t flag;
    uint16_t version;
    uint8_t numCards;
    uint8_t numFingers;
    uint8_t numFaces;
    uint8_t infoMask;
    uint32_t authGroupID;
    uint32_t faceChecksum;
} BS2User;
```

1. *userID*
User ID provided as string, and has a range of 1 ~ 4294967295.

2. *formatVersion*
Not Used.

3. *flag*
Flag that shows the user's status. OR operation is available and the mask value is listed below.

Value	Description
0x00	None
0x01	User enrolled
0x02	User updated
0x04	User deleted
0x80	User disabled

4. *version*
Not Used.

5. *numCards*
Number of cards mapped to user.

6. *numFingers*
Number of fingerprint templates mapped to user.

7. *numFaces*
Number of face templates mapped to user.

8. *infoMask*
[+ 2.8.3] This indicates what information the user has. By changing the information at infoMask and sending the information to change to the device, it is possible to partially change the selected user’s information.

At this moment, Partial Update families of functions(BS2_PartialUpdateUser, BS2_PartialUpdateUserEx, BS2_PartialUpdateUserSmall, BS2_PartialUpdateUserSmallEx, BS2_PartialUpdateUserFaceEx) and user mask argument must match (refer to the table below).

Value	Description
0x01	BS2_USER_INFO_MASK_PHRASE
0x02	BS2_USER_INFO_MASK_JOB_CODE
0x04	BS2_USER_INFO_MASK_NAME
0x08	BS2_USER_INFO_MASK_PHOTO
0x10	BS2_USER_INFO_MASK_PIN
0x20	BS2_USER_INFO_MASK_CARD
0x40	BS2_USER_INFO_MASK_FINGER
0x80	BS2_USER_INFO_MASK_FACE

Get user information
The infoMask indicates what information is allocated to the current user when getting the user information.

User information	BS2_USER_MASK	infoMask
Partial removal	unmasking	unmasking
Partial edit	masking	masking
Default setting	unmasking	masking

Partial removal of user information
Choose unmasking for the information to be removed at all sections (infoMask, Partial Update families of functions, and user mask agreements).

Partial edit of user information
Choose masking for the information to be edited at all sections (infoMask, Partial Update families of functions, and user mask agreements).

Default setting of user information
Choose masking for infoMask and unmasking for Partial Update familes of functions and user mask agreements.

Changing credential information (Card/Fingerprint/Face)
It checks if the credential information is allocated or not such as [numCards, numFingers, or numFaces is 0], [fingerObjs, cardObjs, faceObjs, or faceExObjs is NULL]. Plus, it checks infoMask.
If the input information at card/fingerprint/face is greater than 0 and the mask at infoMask is set, the device credential information can be changed.
For example, when the device has two fingerprints for a user, select masking at infoMask's BS2_USER_INFO_MASK_FINGER, numFingers =1, and assign a fingerprint at fingerObjs. Then the device will have only one newly assigned fingerprint.
To add a new fingerprint, three fingerprints must be assigned which includes the two previously stored fingerprints and the new fingerprint.

Keeping the credential information (Card/Fingerprint/Face)
The device keeps the original credential information if the credential information at card/fingerprint/face is set to 0 and the infoMask is set to masking.

Removing the credential information (Card/Fingerprint/Face)
If the credential information at card/fingerprint/face is set to 0 and the infoMask is set to unmasking, the device erases the corresponding information assigned to each credential.

9. *authGroupID*
The group ID the user is assigned to when face group matching is enabled.

10. *faceChecksum*
Not Used.

BS2UserSetting

> **Tip**
> Please use BS2UserSettingEx to set **the personal authentication mode** for **Visual Face-based devices**.

```
typedef struct {
    uint32_t startTime;
    uint32_t endTime;
    uint8_t fingerAuthMode;
    uint8_t cardAuthMode;
    uint8_t idAuthMode;
    uint8_t securityLevel;
} BS2UserSetting;
```

1. *startTime*
Start time that a user can identify.
A value greater than 978307200 (01 January 2001, 00:00:00) must be entered, **If set 0**, it means that there is **no limit**.

2. *endTime*
End time that that a user can identify.
A value smaller than 1924991999 (31 December 2030, 23:59:59) must be entered, **If set 0**, it means that there is **no limit**.

For firmware versions that include the date extension feature or later, settings can be stored up to 2145916799(31 December 2037, 23:59:59).
Please refer to the revision notes for each firmware version to confirm whether this is supported.
(For example, in the case of BioStation 3, firmware version 1.3.0 or later allows settings up to 2145916799(31 December 2037, 23:59:59).)

3. *fingerAuthMode*
Finger authentication mode for user authentication.

Value	Description
0	Uses only fingerprint authentication
1	Uses fingerprint and PIN authentication
254	Cannot use
255	Undefined(Operates as defined in system)

4. *cardAuthMode*

Card authentication mode for user authentication.

Value	Description
2	Uses only card authentication
3	Uses card and fingerprint authentication
4	Uses card and PIN authentication
5	Uses fingerprint or PIN after card authentication
6	Uses card, fingerprint, and PIN authentication
254	Cannot use
255	Undefined(Operates as defined in system)

5. *idAuthMode*

ID authentication mode for user authentication.

Value	Description
7	Uses fingerprint authentication after entering user ID
8	Uses PIN authentication after entering user ID
9	Uses fingerprint or PIN authentication after entering user ID
10	Uses fingerprint and PIN authentication after entering user ID
254	Cannot use
255	Undefined(Operates as defined in system)

6. *securityLevel*

Security level for fingerprint identification or face recognition.

Value	Description
0	Default value defined in system
1	Lowest security level
2	Low security level
3	Normal security level
4	High security level
5	Highest security level

BS2UserPhoto

```
typedef struct {
    uint32_t size;
    uint8_t data[BS2_USER_PHOTO_SIZE];
} BS2UserPhoto;
```

1. *size*
Size of the user profile image data.

2. *data*
Data of the profile image, which can be stored up to 16kb.

BS2UserBlob

```
typedef struct {
    BS2User user;
    BS2UserSetting setting;
    uint8_t name[BS2_USER_NAME_SIZE];
    BS2UserPhoto photo;
    uint8_t pin[BS2_PIN_HASH_SIZE];
    BS2CSNCard* cardObjs;
    BS2Fingerprint* fingerObjs;
    BS2Face* faceObjs;
    uint32_t accessGroupId[BS2_MAX_NUM_OF_ACCESS_GROUP_PER_USER];
} BS2UserBlob;
```

1. *user*
Structure that defines the basic user information.

2. *setting*
Structure that defines the configuration value for user identification.

3. *name*
User name having UTF-8 for string encoding.

4. *photo*
User profile image, which supports only Jpeg images.

5. *pin*
Personal Identification Number(PIN). It should be entered through *BS_MakePinCode* function.

6. *cardObjs*
Card list for user authentication that needs to exist as much as **user.numCards**. Refer to Smartcard API for data format.
In case of Secure Crendential card, cardObjs array of BS2UserBlob structure should be filled and the user should be updated after Secure Crendential card issuing.

7. *fingerObjs*

Fingerprint template for user authentication that needs to exist as much as **user.numFingers**. Refer to Fingerprint API for data format.

8. *faceObjs*
Face template for user authentication that needs to exist as much as **user.numFaces**. Refer to Face API for data format.

9. *accessGroupId*
List of access groups where users belong to which can be configured up to 16 groups.

BS2Job

```
typedef struct {
    uint8_t numJobs;
    uint8_t reserved[3];

    struct {
        BS2_JOB_CODE code;
        BS2_JOB_LABEL label;
    } jobs[BS2_MAX_JOB_SIZE];
} BS2Job;
```

1. *numJobs*
Number of job codes allocated to the user.

2. *reserved*
Reserved Space.

3. *jobs*
List of jobs.

BS2UserBlobEx

```
typedef struct {
    BS2User user;
    BS2UserSetting setting;
    uint8_t name[BS2_USER_NAME_SIZE];
    BS2UserPhoto photo;
    uint8_t pin[BS2_PIN_HASH_SIZE];
    BS2CSNCard* cardObjs;
    BS2Fingerprint* fingerObjs;
    BS2Face* faceObjs;
    BS2Job job;
    BS2_USER_PHRASE phrase;
    uint32_t accessGroupId[BS2_MAX_NUM_OF_ACCESS_GROUP_PER_USER];
} BS2UserBlobEx;
```

1. *user*
Structure that defines the basic user information.

2. *setting*
Structure that defines the configuration value for user identification.

3. *name*
User name having UTF-8 for string encoding.

4. *photo*
User profile image, which supports only Jpeg images.

5. *pin*
Personal Identification Number(PIN). It should be entered through *BS_MakePinCode* function.

6. *cardObjs*
Card list for user authentication that needs to exist as much as **user.numCards**. Refer to Smartcard API for data format.

7. *fingerObjs*
Fingerprint template for user authentication that needs to exist as much as **user.numFingers**. Refer to Fingerprint API for data format.

8. *faceObjs*
Face template for user authentication that needs to exist as much as **user.numFaces**. Refer to Face API for data format.

9. *job*
Job code that will be allocated to user.

10. *phrase*
Private message that will be displayed when the user authenticates.

Model	Supported Ver.
FaceStation 2	V1.0.0 or higher
FaceStation F2	V1.0.0 or higher
X-Station 2	V1.0.0 or higher
BioStation 3	V1.0.0 or higher

11. *accessGroupId*
List of access groups where users belong to which can be configured up to 16 groups.

BS2UserSmallBlob

```
typedef struct {
    BS2User user;
    BS2UserSetting setting;
    uint8_t name[BS2_USER_NAME_SIZE];
    BS2UserPhoto* photo;
    uint8_t pin[BS2_PIN_HASH_SIZE];
    BS2CSNCard* cardObjs;
    BS2Fingerprint* fingerObjs;
    BS2Face* faceObjs;
    uint32_t accessGroupId[BS2_MAX_NUM_OF_ACCESS_GROUP_PER_USER];
} BS2UserSmallBlob;
```

1. *user*
Structure that defines the basic user information.

2. *setting*
Structure that defines the configuration value for user identification.

3. *name*
User name having UTF-8 for string encoding.

4. *photo*
User profile image, which supports only Jpeg images.

5. *pin*
Personal Identification Number(PIN). It should be entered through *BS_MakePinCode* function.

6. *cardObjs*
Card list for user authentication that needs to exist as much as **user.numCards**. Refer to Smartcard API for data format.

7. *fingerObjs*
Fingerprint template for user authentication that needs to exist as much as **user.numFingers**. Refer to Fingerprint API for data format.

8. *faceObjs*
Face template for user authentication that needs to exist as much as **user.numFaces**. Refer to Face API for data format.

9. *accessGroupId*
List of access groups where users belong to which can be configured up to 16 groups.

BS2UserSmallBlobEx

```
typedef struct {
    BS2User user;
    BS2UserSetting setting;
    uint8_t name[BS2_USER_NAME_SIZE];
```

```
    BS2UserPhoto* photo;
    uint8_t pin[BS2_PIN_HASH_SIZE];
    BS2CSNCard* cardObjs;
    BS2Fingerprint* fingerObjs;
    BS2Face* faceObjs;
    BS2Job job;
    BS2_USER_PHRASE phrase;
    uint32_t accessGroupId[BS2_MAX_NUM_OF_ACCESS_GROUP_PER_USER];
} BS2UserSmallBlobEx;
```

1. *user*
Structure that defines the basic user information.

2. *setting*
Structure that defines the configuration value for user identification.

3. *name*
User name having UTF-8 for string encoding.

4. *photo*
User profile image, which supports only Jpeg images.

5. *pin*
Personal Identification Number(PIN). It should be entered through *BS_MakePinCode* function.

6. *cardObjs*
Card list for user authentication that needs to exist as much as **user.numCards**. Refer to Smartcard API for data format.

7. *fingerObjs*
Fingerprint template for user authentication that needs to exist as much as **user.numFingers**. Refer to Fingerprint API for data format.

8. *faceObjs*
Face template for user authentication that needs to exist as much as **user.numFaces**. Refer to Face API for data format.

9. *job*
Job code that will be allocated to user.

10. *phrase*
Private message that will be displayed when the user authenticates.

Model	Supported Ver.
FaceStation 2	V1.0.0 or higher
FaceStation F2	V1.0.0 or higher
X-Station 2	V1.0.0 or higher

11. *accessGroupId*
List of access groups where users belong to which can be configured up to 16 groups.

BS2UserSettingEx

> **Tip**
> Please use BS2UserSetting to set **the personal authentication mode** for devices other than **Visual Face-based devices**.

```
typedef struct {
    uint8_t faceAuthMode;
    uint8_t fingerprintAuthMode;
    uint8_t cardAuthMode;
    uint8_t idAuthMode;
    uint8_t reserved[28];
} BS2UserSettingEx;
```

1. *faceAuthMode*
User facial authentication mode

Value	Level 1	Level 2	Level 3	Level 4
11	Face			
12	Face	Fingerprint		
13	Face	PIN		
14	Face	Fingerprint or PIN		
15	Face	Fingerprint	PIN	
254	Cannot use			
255	Not defined(System defined mode)			

2. *fingerprintAuthMode*
User fingerprint authentication mode

Value	Level 1	Level 2	Level 3	Level 4
16	Fingerprint			
17	Fingerprint	Face		
18	Fingerprint	PIN		
19	Fingerprint	Face or PIN		
20	Fingerprint	Face	PIN	
254	Cannot use			

Value	Level 1	Level 2	Level 3	Level 4
255	Not defined(System defined mode)			

3. *cardAuthMode*
User card authentication mode

Value	Level 1	Level 2	Level 3	Level 4
21	Card			
22	Card	Face		
23	Card	Fingerprint		
24	Card	PIN		
25	Card	Face or Fingerprint		
26	Card	Face or PIN		
27	Card	Fingerprint or PIN		
28	Card	Face or Fingerprint or PIN		
29	Card	Face	Fingerprint	
30	Card	Face	PIN	
31	Card	Fingerprint	Face	
32	Card	Fingerprint	PIN	
33	Card	Face or Fingerprint	PIN	
34	Card	Face	Fingerprint or PIN	
35	Card	Fingerprint	Face or PIN	
254	Cannot use			
255	Not defined(System defined mode)			

4. *idAuthMode*
User ID authentication mode

Value	Level 1	Level 2	Level 3	Level 4
36	ID	Face		
37	ID	Fingerprint		
38	ID	PIN		
39	ID	Face or Fingerprint		
40	ID	Face or PIN		
41	ID	Fingerprint or PIN		

Value	Level 1	Level 2	Level 3	Level 4
42	ID	Face or Fingerprint or PIN		
43	ID	Face	Fingerprint	
44	ID	Face	PIN	
45	ID	Fingerprint	Face	
46	ID	Fingerprint	PIN	
47	ID	Face or Fingerprint	PIN	
48	ID	Face	Fingerprint or PIN	
49	ID	Fingerprint	Face or PIN	
254	Cannot use			
255	Not defined(System defined mode)			

5. *reserved*
Reserved

BS2UserFaceExBlob

```
typedef struct
{
    BS2User user;
    BS2UserSetting setting;
    BS2_USER_NAME user_name;
    BS2UserPhoto* user_photo_obj;
    BS2_USER_PIN pin;
    BS2CSNCard* cardObjs;
    BS2Fingerprint* fingerObjs;
    BS2Face* faceObjs;                  // FS2, FL
    BS2Job job;
    BS2_USER_PHRASE phrase;
    BS2_ACCESS_GROUP_ID accessGroupId[BS2_MAX_NUM_OF_ACCESS_GROUP_PER_USER];

    BS2UserSettingEx settingEx;         // F2
    BS2FaceEx* faceExObjs;              // F2
} BS2UserFaceExBlob;
```

1. *user*
Basic user information defined structure

2. *setting*
Basic user setting defined structure

3. *name*
User name (Encoding : UTF-8)

4. *photo*
User profile image (Only support jpeg)

5. *pin*
PIN, must be filled with a return of API *BS2_MakePinCode* 6. *cardObjs*
Card list for user authentication, there must be as many as **user.numCards**. Refer to Smartcard API for data format.

7. *fingerObjs*
Fingerprint template list for user authentication, there must be as many as **user.numFingers**. Refer to Fingerprint API for data format.

8. *faceObjs*
`IR Face supported` Face template list for user authentication, there must be as many as **user.numFaces**. Refer to Face API for data format.

9. *job*
Job code in T&A mode

10. *phrase*
Private message that will be displayed when the user authenticates.

Model	**Supported Ver.**
FaceStation 2	V1.0.0 or higher
FaceStation F2	V1.0.0 or higher
X-Station 2	V1.0.0 or higher

11. *accessGroupId*
List of access group of the user assigned, maximum is 16.

12. *settingEx*
`Visual Face supported` Sets private authentication. It is now possible to combine more various authentication modes by combining fingerprints and faces.

13. *faceExObjs*
`Visual Face supported` Face template list for user authentication, there must be as many as **user.numFaces**. Refer to Face API for data format.

BS2UserStatistic

```
typedef struct {
    uint32_t numUsers;
```

```
    uint32_t numCards;
    uint32_t numFingerprints;
    uint32_t numFaces;
    uint32_t numNames;
    uint32_t numImages;
    uint32_t numPhrases;
} BS2UserStatistic;
```

1. *numUsers*
Number of registered users.

2. *numCards*
Number of registered cards.

3. *numFingerprints*
Number of fingerprints registered.

4. *numFaces*
Number of registered faces.

5. *numNames*
Number of registered user names.

6. *numImages*
Number of images registered.

7. *numPhrases*
Number of registered personal messages.

BS2UserOverride

```
typedef struct{
    BS2_USER_ID userID;                ///< 32 bytes
    BS2_BOOL useExtendedAutoLockTimeout;   ///<  1 byte
    uint8_t reserved[11];          ///< 11 bytes
} BS2UserOverride;
```

1. *userID*
User ID.

2. *useExtendedAutoLockTimeout*
Indicates whether the user uses the extended door open time (user override).

3. *reserved*
Reserved space.