Table of Contents

Quick Guide 1
Step 1. Initializing the SDK 1
Step 2. Connecting the device 2
Step 3. Verifying the device function 3
Step 4. Getting the device configuration 3
Step 5. Enrolling a new user 4
User header 4
Enroll a card 5
Add the fingerprint template 6
Add the face template 8
Step 6. Managing the log 8
Getting the log 9
Open Source License 9

Quick Guide

This page explains how to develop an application using the BioStar SDK.
The content below delineates the instructions of the interface that is used frequently by the BioStar application and it has been written based on C++. Refer to the Example folder of the SDK package for examples in other languages.

Parameters having variable name that ends with Obj allocates the memory dynamically. To deallocate the memory, use the BS2_ReleaseObject function at the end of its usage. Functions that needs deallocation of the memory are listed below.

```
int BS2_GetDevices(void* context, BS2_DEVICE_ID** deviceListObj,
uint32_t* numDevice);
int BS2_GetLog(void* context, BS2_DEVICE_ID deviceId, BS2_EVENT_ID
eventId, uint32_t amount, BS2Event** logsObj, uint32_t* numLog);
int BS2_GetFilteredLog(void* context, BS2_DEVICE_ID deviceId, char* uid,
BS2_EVENT_CODE eventCode, BS2_TIMESTAMP start, BS2_TIMESTAMP end, uint8_t
tnakey, BS2Event** logsObj, uint32_t* numLog);
int BS2_GetUserList(void* context, BS2_DEVICE_ID deviceId, char**
uidsObj, uint32_t* numUid);
```

Step 1. Initializing the SDK

To use the SDK, the Context must be created and initialized. The BS_SDK_ERROR_UNINITIALIZED error will be returned, when calling other functions without the initializing process.

```
int main(int argc, char* argv[])
{
    void* context = NULL;

    context = BS2_AllocateContext();
    if(context != NULL)
    {
        int result = BS2_Initialize(context);
        if(result == BS_SDK_SUCCESS)
        {
            // do something ...
        }
    }
    else
    {
        printf("Out of memory\n");
    }

    if(context != NULL)
```

```
    {
        BS2_ReleaseContext(context);
    }
    return ;
}
```

Step 2. Connecting the device

Server mode and *direct mode* can be used as methods to connect the BioStar application with the device. *Server mode* is a way to connect by the device sending signals to the BioStar application, and *direct mode* is a way to connect by sending signals to the device from the BioStar application. The devices are set to *direct mode* as default, and connecting as *direct mode* is demonstrated below.

- When knowing the IP address and Port number.

```
    const char* deviceAddress = "192.168.1.2";
    uint16_t devicePort = 51211;
    uint32_t deviceId = ;
    int result = BS2_ConnectDeviceViaIP(context, deviceAddress,
devicePort, &deviceId);
    if(result == BS_SDK_SUCCESS)
    {
        printf("The device ID while connected to the network is
%d\n", deviceId);
    }
    else
    {
        printf("Failed to connect to device. (error code : 0x%x)\n",
result);
    }
```

- When connecting the device after searching

```
    uint32_t* deviceListObj = NULL;
    uint32_t numDevice = ;
    uint32_t selectedDeviceId = ;
    int result = BS2_SearchDevices(context);
    if(result == BS_SDK_SUCCESS)
    {
        result = BS2_GetDevices(context, &deviceListObj,
&numDevice);
        if(result == BS_SDK_SUCCESS)
        {
            // TODO select proper device id
            selectedDeviceId = deviceListObj[];
```

```
            // free device list object
            BS2_ReleaseObject(deviceListObj);

            result = BS2_ConnectDevice(context, selectedDeviceId);
        }
    }
```

Step 3. Verifying the device function

Once the connection with the device has been completed, it's required to get the information of the device. The design of the BioStar application's UI needs to be oriented with a certain type of device because some functions are not supported based on the device's type.[1] Use BS2_GetDeviceInfo to get the device's information.

```
uint32_t deviceId = 1;
BS2SimpleDeviceInfo deviceInfo;
int result = BS2_GetDeviceInfo(context, deviceId, &deviceInfo);
if(result == BS_SDK_SUCCESS)
{
    //TODO Customizing the UI
}
```

Step 4. Getting the device configuration

It will be able to get the configuration information by using the BS2_GetXXXConfig[2] function. Refer to Configuration API for more specific information about the device's configuration.

```
uint32_t deviceId = 1;
BS2SimpleDeviceInfo deviceInfo;
BS2TNAConfig tnaConfig;
BS2IpConfig ipconfig;

int result = BS2_GetIPConfig(context, deviceId, &ipconfig);
if(result == BS_SDK_SUCCESS)
{
    //TODO handle it
}

if(deviceInfo.tnaSupported)
{
    result = BS2_GetTNAConfig(context, deviceId, &tnaConfig);
    if(result == BS_SDK_SUCCESS)
    {
        //TODO handle it
```

```
    }
}
```

Step 5. Enrolling a new user

To enroll a new user to the device, it is required to set the header information properly. The type of credentials that a user can use are PIN, smart card, finger, face, and use the information from Step 4. Getting the device configuration `BS2SimpleDeviceInfo` to be informed on which credential the device supports.

The user and credential information can be enrolled by the `BS2_EnrolUser` function. For more details, refer to User Management API.

```
uint32_t deviceId = 1;
BS2UserBlob userBlob;

//TODO fill up user header
int result = BS2_EnrolUser(context, deviceId, &userBlob);
if(result != BS_SDK_SUCCESS)
{
    //TODO handle error
}
```

User header

The user header information is different depending on the type of device. For instance, BioStation 2 can use the user identifier, user name, and PIN, but BioEntry Plus can only use the user identifier field. Because the rest of the fields that are not used from the device will be ignored, a BioStar application developer must create a UI based on the function that the device supports to avoid users from experiencing inconvenience and confusion.

> **NOTE**
> User identifier must be a number and the valid range is 1 ~ 4294967295.

```
uint32_t deviceId = 1;
BS2SimpleDeviceInfo deviceInfo;
BS2UserBlob userBlob;

memset(&userBlob, , sizeof(BS2UserBlob));

//setup user id
strcpy(userBlob.user.userID, "user1");
```

```
userBlob.setting.startTime = time(NULL);
userBlob.setting.endTime = userBlob.setting.startTime + 7*24*60*60;
// 1 week
userBlob.setting.idAuthMode = BS2_AUTH_MODE_NONE;
userBlob.setting.securityLevel = BS2_USER_SECURITY_LEVEL_DEFAULT;

if(deviceInfo.cardSupported)
{
    userBlob.setting.cardAuthMode = BS2_AUTH_MODE_CARD_ONLY;
}
else
{
    userBlob.setting.cardAuthMode = BS2_AUTH_MODE_NONE;
}

if(deviceInfo.fingerSupported)
{
    userBlob.setting.fingerAuthMode = BS2_AUTH_MODE_BIOMETRIC_ONLY;
}
else
{
    userBlob.setting.fingerAuthMode = BS2_AUTH_MODE_NONE;
}

if(deviceInfo.userNameSupported)
{
    strcpy(userBlob.user_name, "Joshua");
}

if(deviceInfo.pinSupported)
{
    const char* plaintext = "my password";
    int result = BS2_MakePinCode(context, plaintext, userBlob.pin);
    if(result != BS_SDK_SUCCESS)
    {
        //TODO handle error
    }
}
```

Enroll a card

The BioStar system supports various cards, such as MIFARE, IClass, and more, which can be allocated up to 8 cards per person. To allocate a smart card to a user, use the BS2_ScanCard function to read the card information and then map it to the User header structure.

```
uint32_t deviceId = 1;
BS2SimpleDeviceInfo deviceInfo;
```

```
BS2UserBlob userBlob;
BS2Card cardList[BS2_MAX_NUM_OF_CARD_PER_USER];

if(deviceInfo.cardSupported)
{
    int idx = ;
    for(; idx < BS2_MAX_NUM_OF_CARD_PER_USER ; idx++)
    {
        int result = BS2_ScanCard(context, deviceId, cardList + idx,
NULL);
        if(result != BS_SDK_SUCCESS)
        {
            //TODO handle error
            break;
        }
    }

    userBlob.user.numCards = idx;
    userBlob.cardObjs = cardList;
}
```

Add the fingerprint template

Extracting the fingerprint template is done by 3 steps;
1. scanning the fingerprint image
2. extracting the template data
3. matching the enrolled fingerprints

Once the extraction of the fingerprint is completed, map the fingerprint template information to the structure.

```
uint32_t deviceId = 1;
BS2SimpleDeviceInfo deviceInfo;
BS2UserBlob userBlob;
BS2Fingerprint fingerprintList[BS2_MAX_NUM_OF_FINGER_PER_USER];

if(deviceInfo.fingerSupported)
{
    int idx = ;
    uint32_t templateIndex = ;
    uint32_t fingerprintQuality =
BS2_FINGER_TEMPLATE_QUALITY_STANDARD;
    uint8_t templateFormat = BS2_FINGER_TEMPLATE_FORMAT_SUPREMA;
    int result = BS_SDK_SUCCESS;
    for(; idx < BS2_MAX_NUM_OF_FINGER_PER_USER; idx++)
    {
        for(templateIndex =  ; templateIndex <
BS2_TEMPLATE_PER_FINGER ;)
```

```
        {
            result = BS2_ScanFingerprint(context, deviceId,
fingerprintList + idx, templateIndex, fingerprintQuality,
            fingerprintFormat, NULL);
            if(result != BS_SDK_SUCCESS)
            {
                if (result == BS_SDK_ERROR_EXTRACTION_LOW_QUALITY ||
                    result == BS_SDK_ERROR_CAPTURE_LOW_QUALITY)
                {
                    printf("Low quality. try again.\n");
                }
                else
                {
                    //TODO handle error
                    break;
                }
            }
            else
            {
                templateIndex++;
            }
        }

        if(result != BS_SDK_SUCCESS)
        {
            break;
        }
    }

    if(result == BS_SDK_SUCCESS)
    {
        result = BS2_VerifyFingerprint(context, deviceId,
fingerprintList);
        if(result == BS_SDK_SUCCESS)
        {
            userBlob.user.numFingers = idx;
            userBlob.fingerObjs= fingerprintList;
        }
        else
        {
            if(result == BS_SDK_ERROR_NOT_SAME_FINGERPRINT)
            {
                printf("The fingerprint doesn't match.\n");
            }

            //TODO handle error
        }
    }
}
```

Add the face template

Extracting the face template is done by 2 steps;
1. scanning the fingerprint image
2. extracting the template data

Face template doesn't have verify step. Once the extraction of the face template is completed, map the face template information to the structure.

```
uint32_t deviceId = 1;
BS2SimpleDeviceInfo deviceInfo;
BS2UserBlob userBlob;
BS2Face Face[BS2_MAX_NUM_OF_FACE_PER_USER];

if (deviceInfo.faceSupported)
{
    int idx = ;
    uint32_t templateIndex = ;
    byte enrollThreshold;
    int result = BS_SDK_SUCCESS;



    for(; idx < BS2_MAX_NUM_OF_FACE_PER_USER; idx++)
    {
            result = BS2_ScanFace(context, deviceId, Face,
enrollThreshold, NULL);
            if(result != BS_SDK_SUCCESS)
            {
                    //TODO handle error
                    break;
            }
            if(result == BS_SDK_SUCCESS)
            {
                     Face[].faceindext = idx;
                     userBlob.faceObjs= face[]
            }

    }
}
```

Step 6. Managing the log

BioEntry Plus, BioEntry W, BioLite Net, Xpass, Xpass S2 can store up to 50,000 logs, and BioStation 2 can store up to 3,000,000 logs. The log information is managed by a circular

queue, so if there is not enough space to store the old logs will be automatically deleted to secure space. For more details, refer to Log Management API.

Getting the log

Use BS2_GetLog and BS2_GetFilteredLog to get the logs. In most cases, the BS2_GetLog is used, but when search conditions are needed use the BS2_GetFilteredLog

```
uint32_t deviceId = 1;
BS2Event* logs = NULL;
uint32_t numLogs = ;
uint32_t endTime = time(NULL);
uint32_t startTime = endTime - 7*24*60*60; //last week

// Get all logs
int result = BS2_GetLog(context, deviceId, , , &logs, &numLogs);
if(result == BS_SDK_SUCCESS)
{
    uint32_t idx = ;
    for(idx =  ; idx < numLogs ; idx++)
    {
        // TODO handle it
    }

    BS2_ReleaseObject(logs);
}

// Filtering logs
result = BS2_GetFilteredLog(context, deviceId, NULL, , startTime,
endTime, , &logs, &numLogs);
if(result == BS_SDK_SUCCESS)
{
    uint32_t idx = ;
    for(idx =  ; idx < numLogs ; idx++)
    {
        // TODO handle it
    }

    BS2_ReleaseObject(logs);
}
```

Open Source License

BioStar 2 Device SDK uses the “OpenSSL”, which is licensed under the OpenSSL and Original SSLeay licenses. As for the OpenSSL and Original SSLeay licenses, please refer to OpenSSL License

and Original SSLeay License.

1)

For example, Xpass does not have to have fingerprint identification and face recognition functions as a result of not having a fingerprint sensor and a face recognition sensor.

2)

BS2_GetFactoryConfig, BS2_GetSystemConfig, BS2_GetAuthConfig, BS2_GetDisplayConfig, BS2_GetIPConfig, BS2_GetTNAConfig, BS2_GetCardConfig, BS2_GetFingerprintConfig