

BS2_PartialUpdateUserFaceEx 1

..... 1

..... 1

..... 2

..... 2

(C++) 2

(C#) 4

BS2_PartialUpdateUserFaceEx

[+ 2.8.3]

mask
, BS2User infoMask

```
#include "BS_API.h"
```

```
int BS2_PartialUpdateUserFaceEx(void* context, uint32_t deviceId,
BS2_USER_MASK mask, BS2UserFaceExBlob* userBlob, uint32_t userCount);
```

BS2UserFaceExBlob

- [In] *context* : Context
- [In] *deviceId* :
- [In] *mask* : mask

0x0002	(,)
0x0004	
0x0008	
0x0010	PIN
0x0020	
0x0040	
0x0080	
0x0100	
0x0200	
0x0400	
0x0800	(FSF2, BS3)
0x1000	(FSF2, BS3)

- [In] *userBlob* :
- [In] *userCount* :

BS_SDK_SUCCESS , 가 .
 BS2_EVENT_USER_UPDATE_PARTIAL_SUCCESS 가 .
 BS2_EVENT_USER_UPDATE_PARTIAL_FAIL 가 .

[BS2_PartialUpdateUser](#)
[BS2_PartialUpdateUserEx](#)
[BS2_PartialUpdateUserSmall](#)
[BS2_PartialUpdateUserSmallEx](#)
[BS2_PartialUpdateUserFaceEx](#)

(C++)

[sample_partialupdateuserfaceex.cpp](#)

```

BS2_USER_MASK maskWantUpdate = BS2_USER_MASK_SETTING |
BS2_USER_MASK_SETTING_EX | BS2_USER_MASK_JOB;
int sdkResult = BS_SDK_SUCCESS;

BS2UserFaceExBlob userBlob = { , };
BS2User& user = userBlob.user;
BS2UserSetting& setting = userBlob.setting;
BS2UserSettingEx& settingEx = userBlob.settingEx;

setting.fingerAuthMode = BS2_AUTH_MODE_NONE;
setting.cardAuthMode = BS2_AUTH_MODE_NONE;
setting.idAuthMode = BS2_AUTH_MODE_NONE;

settingEx.faceAuthMode = BS2_AUTH_MODE_NONE;
settingEx.fingerprintAuthMode = BS2_AUTH_MODE_NONE;
settingEx.cardAuthMode = BS2_AUTH_MODE_NONE;
settingEx.idAuthMode = BS2_AUTH_MODE_NONE;

if (BS_SDK_SUCCESS != (sdkResult = uc.getUserBlobUserID(user)))
    return sdkResult;

if ((maskWantUpdate & BS2_USER_MASK_SETTING) == BS2_USER_MASK_SETTING)
{
    if (BS_SDK_SUCCESS != (sdkResult =
uc.getUserBlobExpiryDate(setting)))
        return sdkResult;

    if (BS_SDK_SUCCESS != (sdkResult =
uc.getUserBlobPrivateAuthMode(setting, deviceInfo, deviceInfoEx)))

```

```
        return sdkResult;

        if (BS_SDK_SUCCESS != (sdkResult =
uc.getUserBlobSecurityLevel(setting)))
            return sdkResult;
    }

    if ((maskWantUpdate & BS2_USER_MASK_SETTING_EX) ==
BS2_USER_MASK_SETTING_EX)
    {
        if (BS_SDK_SUCCESS != (sdkResult =
uc.getUserBlobPrivateAuthModeEx(settingEx, deviceInfo, deviceInfoEx)))
            return sdkResult;
    }

    // ...

    if ((maskWantUpdate & BS2_USER_MASK_JOB) == BS2_USER_MASK_JOB)
    {
        msg.str("");
        msg << "Do you want to change/delete #" << user.userID << " jobs?
(0:Change, 1>Delete)";
        uint32_t selected = Utility::getInput<uint32_t>(msg.str());
        switch (selected)
        {
            case 0:
                if (BS_SDK_SUCCESS != (sdkResult =
uc.getUserBlobJobCode(userBlob.job)))
                    return sdkResult;
                user.infoMask |= BS2_USER_INFO_MASK_JOB_CODE;
                break;

            case 1:
            default:
                maskWantUpdate &= ~BS2_USER_MASK_JOB;
                break;
        }
    }
    else
    {
        // Keep
        user.infoMask |= BS2_USER_INFO_MASK_JOB_CODE;
    }

    // ...

    user.numCards = ;
    if ((maskWantUpdate & BS2_USER_MASK_CARD) == BS2_USER_MASK_CARD)
    {
        msg.str("");
        msg << "Do you want to change/delete #" << user.userID << " cards?"
```

```

(0:Change, 1:Delete)";
    uint32_t selected = Utility::getInput<uint32_t>(msg.str());
    switch (selected)
    {
    case 0:
        if (BS_SDK_SUCCESS != (sdkResult =
uc.getUserBlobCardInfo(&userBlob.cardObjs, user.numCards, id,
deviceInfo, deviceInfoEx)))
            return sdkResult;
        user.infoMask |= BS2_USER_INFO_MASK_CARD;
        break;

    case 1:
    default:
        // unmasking and numCards = 0;
        maskWantUpdate &= ~BS2_USER_MASK_CARD;
        break;
    }
}
else
{
    // Keep
    user.infoMask |= BS2_USER_INFO_MASK_CARD;
}

sdkResult = BS2_PartialUpdateUserFaceEx(context, id, maskWantUpdate,
&userBlob, 1);
if (BS_SDK_SUCCESS != sdkResult)
{
    TRACE("BS2_PartialUpdateUserFaceEx call failed: %d", sdkResult);
    return sdkResult;
}

```

(C#)

[sample_partialupdateuserfaceex.cs](#)

```

BS2_USER_MASK maskWantUpdate = (BS2_USER_MASK)BS2UserMaskEnum.SETTING |
(BS2_USER_MASK)BS2UserMaskEnum.SETTING_EX |
(BS2_USER_MASK)BS2UserMaskEnum.JOB;

BS2ErrorCode sdkResult = BS2ErrorCode.BS_SDK_SUCCESS;
BS2UserFaceExBlob[] userBlob =
Util.AllocateStructureArray<BS2UserFaceExBlob>(1);

userBlob[0].cardObjs = IntPtr.Zero;
userBlob[0].fingerObjs = IntPtr.Zero;

```

```
userBlob[].faceObjs = IntPtr.Zero;
userBlob[].user_photo_obj = IntPtr.Zero;
userBlob[].faceExObjs = IntPtr.Zero;

userBlob[].setting.fingerAuthMode = (byte)BS2FingerAuthModeEnum.NONE;
userBlob[].setting.cardAuthMode = (byte)BS2CardAuthModeEnum.NONE;
userBlob[].setting.idAuthMode = (byte)BS2IDAuthModeEnum.NONE;

userBlob[].settingEx.faceAuthMode = (byte)BS2ExtFaceAuthModeEnum.NONE;
userBlob[].settingEx.fingerprintAuthMode =
(byte)BS2ExtFingerprintAuthModeEnum.NONE;
userBlob[].settingEx.cardAuthMode = (byte)BS2ExtCardAuthModeEnum.NONE;
userBlob[].settingEx.idAuthMode = (byte)BS2ExtIDAuthModeEnum.NONE;

string userID;
if (BS2ErrorCode.BS_SDK_SUCCESS != (sdkResult = getUserBlobUserID(ref
userBlob[].user, out userID)))
    return;

if ((maskWantUpdate & (BS2_USER_MASK)BS2UserMaskEnum.SETTING) ==
(BS2_USER_MASK)BS2UserMaskEnum.SETTING)
{
    if (BS2ErrorCode.BS_SDK_SUCCESS != (sdkResult =
getUserBlobExpiryDate(ref userBlob[].setting)))
        return;

    if (BS2ErrorCode.BS_SDK_SUCCESS != (sdkResult =
getUserBlobPrivateAuthMode(ref userBlob[].setting)))
        return;

    if (BS2ErrorCode.BS_SDK_SUCCESS != (sdkResult =
getUserBlobSecurityLevel(ref userBlob[].setting)))
        return;
}

if ((maskWantUpdate & (BS2_USER_MASK)BS2UserMaskEnum.SETTING_EX) ==
(BS2_USER_MASK)BS2UserMaskEnum.SETTING_EX)
{
    if (BS2ErrorCode.BS_SDK_SUCCESS != (sdkResult =
getUserBlobPrivateAuthModeEx(ref userBlob[].settingEx)))
        return;
}

// ...

if ((maskWantUpdate & (BS2_USER_MASK)BS2UserMaskEnum.JOB) ==
(BS2_USER_MASK)BS2UserMaskEnum.JOB)
{
    Console.WriteLine("Do you want to change/delete #{0} jobs?
(0:Change, 1:Delete)", userID);
    Console.Write(">> ");
}
```

```
int selected = Util.GetInput();
switch (selected)
{
    case : // Change jobs
        if (BS2ErrorCode.BS_SDK_SUCCESS != (sdkResult =
getUserBlobJobCode(ref userBlob[].job)))
            return;

        userBlob[].user.infoMask |=
(byte)BS2UserInfoMaskEnum.JOB_CODE;
        break;

    case 1: // Delete
    default:
        maskWantUpdate &= ~(BS2_USER_MASK)BS2UserMaskEnum.JOB;
        break;
}
}
else
{
    // Keep
    userBlob[].user.infoMask |= (byte)BS2UserInfoMaskEnum.JOB_CODE;
}

userBlob[].user.numCards = ;
if ((maskWantUpdate & (BS2_USER_MASK)BS2UserMaskEnum.CARD) ==
(BS2_USER_MASK)BS2UserMaskEnum.CARD)
{
    Console.WriteLine("Do you want to change/delete #{0} cards?
(0:Change, 1:Delete)", userID);
    Console.Write(">> ");
    int selected = Util.GetInput();
    switch (selected)
    {
        case : // Change cards
            if (BS2ErrorCode.BS_SDK_SUCCESS != (sdkResult =
getUserBlobCardInfo(sdkContext, deviceID, ref userBlob[].cardObjs, ref
userBlob[].user.numCards)))
                return;

            userBlob[].user.infoMask |= (byte)BS2UserInfoMaskEnum.CARD;
            break;

        case 1: // Delete cards on the device
        default:
            // unmasking and numCards = 0;
            maskWantUpdate &= ~(BS2_USER_MASK)BS2UserMaskEnum.CARD;
            break;
    }
}
else
```

```
{
    // Keep
    userBlob[].user.infoMask |= (byte)BS2UserInfoMaskEnum.CARD;
}

// ...

sdkResult = (BS2ErrorCode)API.BS2_PartialUpdateUserFaceEx(sdkContext,
deviceID, maskWantUpdate, userBlob, (UInt32)numOfUser);
if (BS2ErrorCode.BS_SDK_SUCCESS != sdkResult)
    Console.WriteLine("BS2_PartialUpdateUserFaceEx call failed {0}",
sdkResult);
```

From:

<https://kb.supremainc.com/kbtest/> - **BioStar Device SDK**

Permanent link:

https://kb.supremainc.com/kbtest/doku.php?id=ko:bs2_partialupdateuserfaceex&rev=1664497514

Last update: **2022/09/30 09:25**