목차

BS2_PartialUpdateUserFaceEx

[+ 2.8.3] 이미 등록된 사용자의 부분 정보를 갱신합니다. 갱신하고자 하는 사용자는 이미 등록된 존재하는 사용자이어야 합니다.
갱신하고자 하는 부분 정보는 mask를 이용해 선택적으로 지정할 수 있습니다.
부분 정보를 삭제하고자 하는 경우, BS2User의 infoMask와 조합하여 삭제할 수 있습니다.

함수

```
#include "BS_API.h"

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

> BS2UserFaceExBlob 구조체 보기

파라미터

- [In] *context* : Context
- [In] *deviceId* : 장치 식별자
- [In] *mask* : 갱신하고자 하는 사용자 부분에 대한 mask

값	설명
0x0002	사용자설정(개인인증모드,유효기간)
0x0004	사용자명
0x0008	이미지
0x0010	PIN
0x0020	카드
0x0040	지문
0x0080	얼굴
0x0100	출입그룹
0x0200	작업코드
0x0400	개인메시지
0x0800	얼굴 (FSF2, BS3)
0x1000	사용자설정 (FSF2, BS3)

- [In] *userBlob* : 갱신하고자 하는 사용자 부분 정보
- [In] *userCount* : 사용자 개수

반환값

성공적으로 수행될 경우 BS_SDK_SUCCESS를 반환하고,
BS2_EVENT_USER_UPDATE_PARTIAL_SUCCESS 이벤트가 발생됩니다.
실패했을 경우 상응하는 에러 코드를 반환하고, 장치 발생 오류라면
BS2_EVENT_USER_UPDATE_PARTIAL_FAIL 이벤트가 발생됩니다.

함께 보기

BS2_PartialUpdateUser
BS2_PartialUpdateUserEx
BS2_PartialUpdateUserSmall
BS2_PartialUpdateUserSmallEx
BS2_PartialUpdateUserFaceEx

샘플코드(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)
{
    if (BS_SDK_SUCCESS != (sdkResult =
uc.getUserBlobJobCode(userBlob.job)))
        return sdkResult;

    user.infoMask |= BS2_USER_INFO_MASK_JOB_CODE;
}

// ...

user.numCards = ;
if ((maskWantUpdate & BS2_USER_MASK_CARD) == BS2_USER_MASK_CARD)
{
    // ...
}

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

샘플코드(C#)

sample_partialupdateuserfaceex.cs

```
BS2_USER_MASK mask = (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[].cardObjs = IntPtr.Zero;
userBlob[].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 ((mask & (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 ((mask & (BS2_USER_MASK)BS2UserMaskEnum.SETTING_EX) ==
(BS2_USER_MASK)BS2UserMaskEnum.SETTING_EX)
{
    if (BS2ErrorCode.BS_SDK_SUCCESS != (sdkResult =
getUserBlobPrivateAuthModeEx(ref userBlob[].settingEx)))
        return;
}
```

```
// ...

if ((mask & (BS2_USER_MASK)BS2UserMaskEnum.JOB) ==
(BS2_USER_MASK)BS2UserMaskEnum.JOB)
{
    if (BS2ErrorCode.BS_SDK_SUCCESS != (sdkResult =
getUserBlobJobCode(ref userBlob[].job)))
        return;

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

// ...

userBlob[].user.numCards = ;
if ((mask & (BS2_USER_MASK)BS2UserMaskEnum.CARD) ==
(BS2_USER_MASK)BS2UserMaskEnum.CARD)
{
    // ...
}

sdkResult = (BS2ErrorCode)API.BS2_PartialUpdateUserFaceEx(sdkContext,
deviceID, mask, userBlob, (UInt32)1);
if (BS2ErrorCode.BS_SDK_SUCCESS != sdkResult)
    Console.WriteLine("BS2_PartialUpdateUserFaceEx call failed {0}",
sdkResult);
```