목차

BS2_ExtractTemplateFaceEx

[+ 2.7.1] FaceStation F2 얼굴이 포함된 이미지를 가지고, template을 추출합니다.
AOC 카드에 template을 담기 위해 사용합니다.

함수

```
#include "BS_API.h"

int BS2_ExtractTemplateFaceEx(void* context, uint32_t deviceId, const
uint8_t* imageData, uint32_t imageDataLen, bool isWarped, BS2TemplateEx*
templateEx);
```

> BS2TemplateEx구조체 보기

파라미터

- [In] *context* : Context
- [In] *deviceId* : 장치 식별자
- [In] *imageData* : 얼굴이 포함된 이미지 정보를 가리키는 포인터
- [In] *imageDataLen* : imageData의 길이
- [In] *isWarped* : warp된 이미지인지의 여부
- [Out] *templateEx* : imageData가 가리키는 이미지로 추출된 얼굴 template

반환값

성공적으로 수행될 경우 BS_SDK_SUCCESS를 반환하고, 에러가 발생할 경우 상응하는 에러 코드를 반환합니다.

샘플코드

C++

```
int UserControl::extractTemplateFaceEx(BS2_DEVICE_ID id, BS2TemplateEx&
templateEx)
{
    BS2SimpleDeviceInfoEx deviceInfoEx = { , };

    int sdkResult = BS2_GetDeviceInfoEx(context_, id, NULL, &deviceInfoEx);
```

```
    if (BS_SDK_SUCCESS != sdkResult)
    {
        TRACE("BS2_GetDeviceInfoEx call failed: %d", sdkResult);
        return sdkResult;
    }

    bool faceExScanSupported = (deviceInfoEx.supported &
BS2SimpleDeviceInfoEx::BS2_SUPPORT_FACE_EX_SCAN) ==
        BS2SimpleDeviceInfoEx::BS2_SUPPORT_FACE_EX_SCAN;
    if (faceExScanSupported)
    {
        try
        {
            if (Utility::isYes("Do you want to extract faceEx template from
image?"))
            {
                string imagePath = Utility::getInput<string>("Enter the face
image path and name:");//C:\88withphone.jpg
                uint32_t size = Utility::getResourceSize(imagePath);
                shared_ptr<uint8_t> buffer(new uint8_t[size],
ArrayDeleter<uint8_t>());

                size_t dataOffset = offsetof(BS2FaceEx, rawImageData);
                size_t faceSize = dataOffset + size;
                if (Utility::getResourceFromFile(imagePath, buffer, size))
                {
                    sdkResult = BS2_ExtractTemplateFaceEx(context_, id,
buffer.get(), size, , &templateEx);
                    if (BS_SDK_SUCCESS != sdkResult)
                    {
                        TRACE("BS2_ExtractTemplateFaceEx call failed: %d",
sdkResult);
                        return sdkResult;
                    }
                    print(templateEx);
                }
            }
        }
        catch (const std::exception&)
        {

        }
    }

    return sdkResult;
}
BS2_ReleaseObject(uidObj);
```

C#

```
Console.WriteLine("Do you want to register from image? [y/n]");
```

```
Console.Write(">> ");
if (Util.IsYes())
{
    Console.WriteLine("Enter the face image path and name:");
    Console.Write(">> ");
    string imagePath = Console.ReadLine();

    if (!File.Exists(imagePath))
    {
        Console.WriteLine("Invalid file path");
        return;
    }

    Image faceImage = Image.FromFile(imagePath);
    if
(!faceImage.RawFormat.Equals(System.Drawing.Imaging.ImageFormat.Jpeg))
    {
        Console.WriteLine("Invalid image file format");
        return;
    }

    IntPtr imageData = IntPtr.Zero;
    UInt32 imageLen = ;
    if (Util.LoadBinary(imagePath, out imageData, out imageLen))
    {
        if ( == imageLen)
        {
            Console.WriteLine("Empty image file");
            return;
        }

        int structHeaderSize = Marshal.SizeOf(typeof(BS2FaceExUnwarped));
        int totalSize = structHeaderSize + (int)imageLen;
        userBlob[].faceExObjs = Marshal.AllocHGlobal(totalSize);
        IntPtr curFaceExObjs = userBlob[].faceExObjs;

        BS2FaceExUnwarped unwarped =
Util.AllocateStructure<BS2FaceExUnwarped>();
        unwarped.flag = ;
        unwarped.imageLen = imageLen;

        Marshal.StructureToPtr(unwarped, curFaceExObjs, false);
        curFaceExObjs += structHeaderSize;

        Util.CopyMemory(curFaceExObjs, imageData, imageLen);

        userBlob[].user.numFaces = 1;
        unwarpedMemory = true;
    }
}
```