목차

BS2_GetImageLog

[+ 2.5.0] 이벤트로그 식별자를 이용하여 이미지로그를 가져옵니다.

함수

```
#include "BS_API.h"

int BS2_GetImageLog(void* context, uint32_t deviceId, uint32_t eventId,
uint8_t** imageObj, uint32_t* imageSize);
```

파라미터

- [In] *context* : Context
- [In] *deviceId* : 장치 식별자
- [In] *eventId* : 이벤트로그 식별자
- [Out] *imageObj* : 이미지로그를 저장할 포인터
- [Out] *imageSize* : 이미지로그의 크기

> **참고**
> imageObj 변수는 사용 후 BS2_ReleaseObject 함수를 통해 시스템에 메모리를 반환해야 합니다.

반환값

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

```
int getImageLog(void* context, BS2_DEVICE_ID id, BS2_EVENT_ID eventID,
uint8_t* imageBuf, uint32_t& imageSize)
{
    if (!imageBuf)
        return BS_SDK_ERROR_NULL_POINTER;

    uint8_t* imageObj = NULL;
    uint32_t size();
    int sdkResult = BS2_GetImageLog(context, id, eventID, &imageObj, &size);
    if (BS_SDK_SUCCESS == sdkResult)
    {
        memcpy(imageBuf, imageObj, size);
```

```
        imageSize = size;
        if (imageObj)
            BS2_ReleaseObject(imageObj);
    }

    return sdkResult;
}
```

C#

```
void getImageLog(IntPtr sdkContext, UInt32 deviceID, bool isMasterDevice)
{
    BS2SimpleDeviceInfo deviceInfo;
    int structSize = Marshal.SizeOf(typeof(BS2Event));
    UInt16 imageLogEventCode =
(UInt16)BS2EventCodeEnum.DEVICE_TCP_CONNECTED;
    BS2EventConfig eventConfig = Util.AllocateStructure<BS2EventConfig>();
    eventConfig.numImageEventFilter = 1;
    eventConfig.imageEventFilter[].mainEventCode = (byte)(imageLogEventCode
>> 8);
    eventConfig.imageEventFilter[].scheduleID =
(UInt32)BS2ScheduleIDEnum.ALWAYS;

    Console.WriteLine("Trying to get the device[{0}] information.",
deviceID);
    BS2ErrorCode result = (BS2ErrorCode)API.BS2_GetDeviceInfo(sdkContext,
deviceID, out deviceInfo);
    if (result != BS2ErrorCode.BS_SDK_SUCCESS)
    {
        Console.WriteLine("Can't get device information(errorCode : {0}).",
result);
        return;
    }

    Console.WriteLine("Trying to activate image log.");
    result = (BS2ErrorCode)API.BS2_SetEventConfig(sdkContext, deviceID, ref
eventConfig);
    if (result != BS2ErrorCode.BS_SDK_SUCCESS)
    {
        Console.WriteLine("Got error({0}).", result);
        return;
    }

    Console.WriteLine("Trying to clear log for quick test.");
    result = (BS2ErrorCode)API.BS2_ClearLog(sdkContext, deviceID);
    if (result != BS2ErrorCode.BS_SDK_SUCCESS)
    {
        Console.WriteLine("Got error({0}).", result);
        return;
    }
```

```
    Console.WriteLine("Trying to disconnect device[{0}] for quick test.",
deviceID);
    result = (BS2ErrorCode)API.BS2_DisconnectDevice(sdkContext, deviceID);
    if (result != BS2ErrorCode.BS_SDK_SUCCESS)
    {
        Console.WriteLine("Got error({0}).", result);
        return;
    }

    Thread.Sleep(500); //waiting for socket close

    Console.WriteLine("Trying to connect device[{0}].", deviceID);
    IntPtr ptrIPAddr = Marshal.StringToHGlobalAnsi(new
IPAddress(BitConverter.GetBytes(deviceInfo.ipv4Address)).ToString());
    //result = (BS2ErrorCode)API.BS2_ConnectDeviceViaIP(sdkContext, new
IPAddress(BitConverter.GetBytes(deviceInfo.ipv4Address)).ToString(),
deviceInfo.port, out deviceID);
    result = (BS2ErrorCode)API.BS2_ConnectDeviceViaIP(sdkContext, ptrIPAddr,
deviceInfo.port, out deviceID);
    Marshal.FreeHGlobal(ptrIPAddr);
    if (result != BS2ErrorCode.BS_SDK_SUCCESS)
    {
        Console.WriteLine("Got error({0}).", result);
        return;
    }

    IntPtr outEventLogObjs = IntPtr.Zero;
    UInt32 outNumEventLogs = ;

    result = (BS2ErrorCode)API.BS2_GetLog(sdkContext, deviceID, , 1024, out
outEventLogObjs, out outNumEventLogs);
    if (result != BS2ErrorCode.BS_SDK_SUCCESS)
    {
        Console.WriteLine("Got error({0}).", result);
        return;
    }

    if (outNumEventLogs > )
    {
        IntPtr curEventLogObjs = outEventLogObjs;
        for (int idx = ; idx < outNumEventLogs; idx++)
        {
            BS2Event eventLog =
(BS2Event)Marshal.PtrToStructure(curEventLogObjs, typeof(BS2Event));
            //if (Convert.ToBoolean(eventLog.image))
            bool hasImage = Convert.ToBoolean(eventLog.image &
(byte)BS2EventImageBitPos.BS2_IMAGEFIELD_POS_IMAGE);
            if (hasImage)
            {
                Console.WriteLine("Trying to get image log[{0}].",
eventLog.id);
```

```
                IntPtr imageObj = IntPtr.Zero;
                UInt32 imageSize = ;

                result = (BS2ErrorCode)API.BS2_GetImageLog(sdkContext,
deviceID, eventLog.id, out imageObj, out imageSize);
                if (result != BS2ErrorCode.BS_SDK_SUCCESS)
                {
                    Console.WriteLine("Got error({0}).", result);
                }
                else
                {
                    int written = ;
                    FileStream file = new
FileStream(String.Format("{0}.jpg", eventLog.id), FileMode.Create,
FileAccess.Write);

                    Console.WriteLine("Trying to save image log[{0}].",
eventLog.id);
                    WriteFile(file.Handle, imageObj, (int)imageSize, out
written, IntPtr.Zero);
                    file.Close();

                    if (written != imageSize)
                    {
                        Console.WriteLine("Got error({0}).", result);
                    }
                    else
                    {
                        Console.WriteLine("Successfully saved the image
log[{0}].", eventLog.id);
                        Process.Start(file.Name);
                    }
                }
                break;
            }

            curEventLogObjs = (IntPtr)((long)curEventLogObjs + structSize);
        }

        API.BS2_ReleaseObject(outEventLogObjs);
    }

    eventConfig.numImageEventFilter = ;

    Console.WriteLine("Trying to deactivate image log.");
    result = (BS2ErrorCode)API.BS2_SetEventConfig(sdkContext, deviceID, ref
eventConfig);
    if (result != BS2ErrorCode.BS_SDK_SUCCESS)
    {
        Console.WriteLine("Got error({0}).", result);
```

```
        return;
    }
}
```