

BS2_GetImageLog

.....

.....

.....

.....

1

1

1

1

BS2_GetImageLog

[+ 2.5.0] 가 .

```
#include "BS_API.h"

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

- [In] *context* : Context
- [In] *deviceId* :
- [In] *eventId* :
- [Out] *imageObj* :
- [Out] *imageSize* :

imageObj [BS2_ReleaseObject](#)

 BS_SDK_SUCCESS , 가
. 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;  
}  
}
```

From:

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

Permanent link:

https://kb.supremainc.com/kbtest/doku.php?id=ko:bs2_getimagelog&rev=1640836921

Last update: **2021/12/30 13:02**