

Table of Contents

BS2_DisableDeviceLicense	1
Declaration	1
Parameter	1
Return Value	1
Sample Code(C++)	2
Sample Code(C#)	3

BS2_DisableDeviceLicense

[+ 2.9.1] Disable device licenses by collectively selecting the device connected as master-slave. Deactivation results for each device are returned through `outResultObj` and `outNumOfResult`. This feature is available only on devices that support the device license activation feature, and the devices that support the feature are listed below.

Supported devices	Firmware
XS2-Finger	V1.2.0
XS2-Card	V1.2.0
BS3	V1.1.0

Declaration

```
#include "BS_API.h"

int BS2_DisableDeviceLicense(void* context, uint32_t deviceId, const
BS2LicenseBlob* licenseBlob, BS2LicenseResult** outResultObj, uint32_t*
outNumOfResult);
```

[See BS2LicenseBlob Structure](#)
[See BS2LicenseResult Structure](#)

Parameter

- [In] `context` : Context
- [In] `deviceId` : Device Identifier
- [In] `licenseBlob` : Device license information structure pointer
- [Out] `outResultObj` : Pointer to structure to receive device license deactivation result
- [Out] `outNumOfResult` : Number of device license deactivation result structures

NOTE

The `outResultObj` parameter must be used and then returned memory to the system via the [BS2_ReleaseObject](#) function.

Return Value

If successfully done, BS_SDK_SUCCESS will be returned. If there is an error, the corresponding error code will be returned.

Sample Code(C++)

[sample_bs2_disabledlicencelicense.cpp](#)

```
int deleteDeviceLicense(void* context, BS2_DEVICE_ID id)
{
    DeviceControl dc(context);
    BS2LicenseBlob licenseBlob = { , };
    vector<BS2_DEVICE_ID> deviceIDs;
    vector<BS2LicenseResult> licenseResult;
    int sdkResult = BS_SDK_SUCCESS;

    licenseBlob.licenseType =
    (BS2_LICENSE_TYPE)Utility::getInput<uint32_t>("Enter the license type.
    (0: None, 1: Visual QR)");
    licenseBlob.numOfDevices =
    (uint16_t)Utility::getInput<uint32_t>("How many devices do you want to
    register?");
    if ( < licenseBlob.numOfDevices)
    {
        // Device ID
        for (uint16_t idx = ; idx < licenseBlob.numOfDevices; idx++)
        {
            BS2_DEVICE_ID deviceID =
            (BS2_DEVICE_ID)Utility::getInput<uint32_t>("Enter a device ID:");
            deviceIDs.push_back(deviceID);
        }

        licenseBlob.deviceIDobjs = deviceIDs.data();

        sdkResult = dc.disableDeviceLicense(id, &licenseBlob,
        licenseResult);
        if (BS_SDK_SUCCESS == sdkResult)
            DeviceControl::print(licenseResult);
    }

    return sdkResult;
}

int DeviceControl::disableDeviceLicense(BS2_DEVICE_ID id, const
BS2LicenseBlob* licenseBlob, vector<BS2LicenseResult>& licenseResult)
{
    BS2LicenseResult* result = NULL;
    uint32_t numOfResult = ;
    int sdkResult = BS2_DisableDeviceLicense(context_, id, licenseBlob,
    &result, &numOfResult);
}
```

```
if (BS_SDK_SUCCESS != sdkResult)
{
    TRACE("BS2_DisableDeviceLicense call failed: %d", sdkResult);
    return sdkResult;
}

licenseResult.clear();
for (uint32_t idx = ; idx < numOfResult; idx++)
{
    licenseResult.push_back(result[idx]);
}

return sdkResult;
}
```

Sample Code(C#)

[sample_bs2_disabledlicencelicense.cs](#)

```
BS2LicenseBlob licenseBlob =
Util.AllocateStructure<BS2LicenseBlob>();

Console.WriteLine("Try removing the license");

Console.WriteLine("Enter the license type. (0: None, 1: Visual
QR)");
Console.Write(">>>> ");
licenseBlob.licenseType =
Util.GetInput((UInt16)BS2LicenseType.VISUAL_QR_MASK);

Console.WriteLine("How many devices do you want to remove?");
Console.Write(">>>> ");
licenseBlob.numOfDevices = Util.GetInput((UInt16)1);

if ( < licenseBlob.numOfDevices)
{
    // Device ID
    List<UInt32> listID = new List<UInt32>();
    UInt32 tempID = ;
    for (UInt16 idx = ; idx < licenseBlob.numOfDevices; idx++)
    {
        Console.WriteLine(" Slave device ID #{0}", idx);
        Console.Write(" >> ");
        tempID = (UInt32)Util.GetInput();
        listID.Add(tempID);
    }

    byte[] byteListID =
```

```
listID.SelectMany(BitConverter.GetBytes).ToArray();
    int byteCount = Marshal.SizeOf(typeof(UInt32)) *
licenseBlob.numOfDevices;

    licenseBlob.deviceIDobjs = Marshal.AllocHGlobal(byteCount);
    Marshal.Copy(byteListID, , licenseBlob.deviceIDobjs,
byteCount);

    // License data
    licenseBlob.licenseLen = ;
    licenseBlob.licenseObj = IntPtr.Zero;

    IntPtr resultObj = IntPtr.Zero;
    UInt32 numOfResult = ;

    BS2ErrorCode result =
(BS2ErrorCode)API.BS2_DisableDeviceLicense(sdkContext, deviceID, ref
licenseBlob, out resultObj, out numOfResult);

    if (BS2ErrorCode.BS_SDK_SUCCESS != result)
    {
        Console.WriteLine("Got error({0}).", result);
    }
    else
    {
        IntPtr curResult = resultObj;
        int resultSize = Marshal.SizeOf(typeof(BS2LicenseResult));
        for (UInt32 idx = ; idx < numOfResult; idx++)
        {
            BS2LicenseResult item =
(BS2LicenseResult)Marshal.PtrToStructure(curResult,
typeof(BS2LicenseResult));
            print(item, idx);
            curResult += resultSize;
        }

        API.BS2_ReleaseObject(resultObj);
    }
} // if (0 < licenseBlob.numOfDevices)

Marshal.FreeHGlobal(licenseBlob.deviceIDobjs);
```

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

Permanent link:
https://kb.supremainc.com/kbtest/doku.php?id=en:bs2_disabledevicelicense

Last update: **2023/03/02 16:12**

