

**BS2\_EnableDeviceLicense** ..... 1

..... 1

..... 1

..... 2

(C++) ..... 2

(C#) ..... 3

# BS2\_EnableDeviceLicense

[+ 2.9.1]

가 ,

outResultObj outNumOfResult  
가 ,

XS2-Finger	V1.2.0
XS2-Card	V1.2.0
BS3	V1.1.0

```
#include "BS_API.h"
```

```
int BS2_EnableDeviceLicense(void* context, uint32_t deviceId, const
BS2LicenseBlob* licenseBlob, BS2LicenseResult** outResultObj, uint32_t*
outNumOfResult);
```

BS2LicenseBlob  
BS2LicenseResult

- [In] *context* : Context
- [In] *deviceId* :
- [In] *licenseBlob* :
- [Out] *outResultObj* :
- [Out] *outNumOfResult* :

outResultObj                      BS2\_ReleaseObject

BS\_SDK\_SUCCESS , 가

## (C++)

[sample\\_bs2\\_enabledlicenselicense.cpp](#)

```
int setDeviceLicense(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();

        string pathName = Utility::getLine("Enter the path and name of
        license.");
        licenseBlob.licenseLen = Utility::getResourceSize(pathName);
        shared_ptr<uint8_t> buffer(new uint8_t[licenseBlob.licenseLen],
        ArrayDeleter<uint8_t>());
        if ( < licenseBlob.licenseLen &&
        Utility::getResourceFromFile(pathName, buffer, licenseBlob.licenseLen))
        {
            licenseBlob.licenseObj = buffer.get();

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

```

    }
}

return sdkResult;
}

int DeviceControl::enableDeviceLicense(BS2_DEVICE_ID id, const
BS2LicenseBlob* licenseBlob, vector<BS2LicenseResult>& licenseResult)
{
    BS2LicenseResult* result = NULL;
    uint32_t numOfResult = ;
    int sdkResult = BS2_EnableDeviceLicense(context_, id, licenseBlob,
&result, &numOfResult);
    if (BS_SDK_SUCCESS != sdkResult)
    {
        TRACE("BS2_EnableDeviceLicense call failed: %d", sdkResult);
        return sdkResult;
    }

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

    return sdkResult;
}

```

## (C#)

[sample\\_bs2\\_enabledlicencse.cs](#)

```

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

    Console.WriteLine("Try adding a 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 register?");
    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
    Console.WriteLine("Enter the path and name of license.");
    Console.Write(">>>> ");
    string licensePath = Console.ReadLine();
    if (!File.Exists(licensePath))
    {
        Console.WriteLine("Invalid license Path");
        return;
    }

    if (Util.LoadBinary(licensePath, out licenseBlob.licenseObj,
out licenseBlob.licenseLen))
    {
        IntPtr resultObj = IntPtr.Zero;
        UInt32 numofResult = ;

        BS2ErrorCode result =
(BS2ErrorCode)API.BS2_EnableDeviceLicense(sdkContext, deviceID, ref
licenseBlob, out resultObj, out numofResult);
        Marshal.FreeHGlobal(licenseBlob.licenseObj);

        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 (Util.LoadBinary(licensePath, out
licenseBlob.licenseObj, out licenseBlob.licenseLen))
} // 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=ko:bs2\\_enabledevicelicense](https://kb.supremainc.com/kbtest/doku.php?id=ko:bs2_enabledevicelicense)

Last update: **2023/03/02 15:54**