

BS2_UpdateOsdpStandardDevice 1

..... 1

..... 1

..... 1

..... 1

(C++) 2

(C#) 3

BS2_UpdateOsdpStandardDevice

[+ 2.8.4] CoreStation40 OSDP , OSDP
outResultObj outNumOfResult

```
#include "BS_API.h"

int BS2_UpdateOsdpStandardDevice(void* context, uint32_t deviceId, const
BS2OsdpStandardDeviceUpdate* osdpDevices, uint32_t numOfDevice,
BS2OsdpStandardDeviceResult** outResultObj, uint32_t* outNumOfResult);
```

BS2OsdpStandardDeviceUpdate BS2OsdpStandardDeviceResult

- [In] *context* : Context
- [In] *deviceId* :
- [In] *osdpDevices* : OSDP
- [In] *numOfDevice* : osdpDevices
- [Out] *outResultObj* :
- [Out] *outNumOfResult* :

outResultObj BS2_ReleaseObject

BS_SDK_SUCCESS , 가

BS2_AddOsdpStandardDevice
BS2_GetOsdpStandardDevice
BS2_GetAvailableOsdpStandardDevice
BS2_UpdateOsdpStandardDevice
BS2_RemoveOsdpStandardDevice
BS2_GetOsdpStandardDeviceCapability
BS2_SetOsdpStandardDeviceSecurityKey

(C++)

sample_bs2_updateosdpstandarddevice.cpp

```
BS2osdpStandardConfig config = { , };
vector<BS2osdpStandardDeviceUpdate> updateData;

BS2_DEVICE_ID id = Utility::selectDeviceID(deviceList, false, false);
int sdkResult = cc.getOsdpStandardConfig(id, config);
if (BS_SDK_SUCCESS != sdkResult)
    return sdkResult;

uint32_t numOfActivated = cc.printOSDPDeviceID(config);
uint32_t numOfDevice = Utility::getInput<uint32_t>("How many devices do
you want to update? (0~%u)", numOfActivated);
if ( < numOfDevice)
{
    for (uint32_t idx = ; idx < numOfDevice; idx++)
    {
        BS2osdpStandardDeviceUpdate item = { , };
        item.deviceID =
(BS2_DEVICE_ID)Utility::getInput<uint32_t>("[%u] Please enter the slave
ID to be updated.", idx + 1);

        if (!ConfigControl::getOsdpID(config, item.deviceID,
item.osdpID))
        {
            cout << "The OSDP ID could not be found." << endl;
            return BS_SDK_ERROR_INTERNAL;
        }

        if (Utility::isYes("Do you want to change the OSDP ID?
(CurrentID: %u)", item.osdpID))
        {
            item.osdpID = (uint8_t)Utility::getInput<uint32_t>("Please
enter the OSDP ID. [0 ~ 126]");
        }

        item.activate = Utility::isYes("Do you like to enable the OSDP
device?");
        item.useSecureSession = Utility::isYes("Does the OSDP device
```

```

use secure communication?");
    item.deviceType = BS2_DEVICE_TYPE_3RD_OSDP_DEVICE;

    updateData.push_back(item);
}

vector<BS20sdpStandardDeviceResult> listResult;
BS20sdpStandardDeviceResult* outResultObj = NULL;
uint32_t outNumOfResult();
int sdkResult = BS2_UpdateOsdpStandardDevice(context_, id,
const_cast<BS20sdpStandardDeviceUpdate*>(updateData.data()),
updateData.size(), &outResultObj, &outNumOfResult);
if (BS_SDK_SUCCESS != sdkResult)
{
    printf("BS2_UpdateOsdpStandardDevice call failed: %d",
sdkResult);
}

if (outResultObj)
{
    listResult.clear();
    for (uint32_t idx = ; idx < outNumOfResult; idx++)
    {
        listResult.push_back(outResultObj[idx]);
    }

    BS2_ReleaseObject(outResultObj);
}
}

return sdkResult;

```

(C#)

[sample_bs2_updateosdpstandarddevice.cs](#)

```

BS20sdpStandardConfig config;
if (!CommonControl.getOsdpStandardConfig(sdkContext, deviceID, out
config))
    return;

UInt32 numOfActivated = ;
printOSDPDeviceID(ref config, ref numOfActivated);

string tempStr = String.Format("How many devices do you want to update?
(0~{0})", numOfActivated);
Util.HighlightLineMulti(tempStr, "How many", "update");
Console.Write(">>>> ");

```

```

int numOfDevice = Util.GetInput(1);
if ( < numOfDevice)
{
    BS2osdpStandardDeviceUpdate[] updateData =
Util.AllocateStructureArray<BS2osdpStandardDeviceUpdate>(numOfDevice);
    for (int idx = ; idx < numOfDevice; idx++)
    {
        tempStr = String.Format(">>>> [{0}] Please enter the slave ID
to be updated.", idx + 1);
        Util.HighlightLine(tempStr, "device ID to be updated");
        Console.Write(">>>> ");
        updateData[idx].deviceID = Util.GetInput((UInt32));

        if (!getOsdpID(ref config, updateData[idx].deviceID, ref
updateData[idx].osdpID))
        {
            Console.WriteLine("The OSDP ID could not be found.");
            return;
        }

        Console.WriteLine("Do you want to change the OSDP ID?
(CurrentID: {0}) [Y/n]", updateData[idx].osdpID);
        Console.Write(">>>> ");
        if (Util.IsYes())
        {
            Util.HighlightLine(">>>> Please enter the OSDP ID. [0 ~
126]", "OSDP ID");
            Console.Write(">>>> ");
            updateData[idx].osdpID = Util.GetInput((byte));
        }

        Util.HighlightLine(">>>> Do you like to enable the OSDP device?
[Y/n]", "enable");
        Console.Write(">>>> ");
        updateData[idx].activate = Convert.ToByte(Util.IsYes());

        Util.HighlightLine(">>>> Does the OSDP device use secure
communication? [Y/n]", "use secure communication");
        Console.Write(">>>> ");
        updateData[idx].useSecureSession =
Convert.ToByte(Util.IsYes());
        updateData[idx].deviceType =
Convert.ToByte(BS2DeviceTypeEnum.THIRD_OSDP_DEVICE);
    }

    List<BS2osdpStandardDeviceResult> listResult = new
List<BS2osdpStandardDeviceResult>();
    int structSize =
Marshal.SizeOf(typeof(BS2osdpStandardDeviceUpdate));
    IntPtr ptrArray = Marshal.AllocHGlobal(structSize *
updateData.Length);

```

```
long ptrCurrent = ptrArray.ToInt64();
BS2ErrorCode result = BS2ErrorCode.BS_SDK_SUCCESS;
try
{
    for (int idx = ; idx < updateData.Length; idx++)
    {
        IntPtr ptrTemp = new IntPtr(ptrCurrent);
        Marshal.StructureToPtr(updateData[idx], ptrTemp, false);
        ptrCurrent += structSize;
    }

    IntPtr outResultObj = IntPtr.Zero;
    UInt32 numOfResult = ;
    result =
    (BS2ErrorCode)API.BS2_UpdateOsdpStandardDevice(sdkContext, deviceID,
ptrArray, (UInt32)updateData.Length, out outResultObj, out
numOfResult);
    if (result != BS2ErrorCode.BS_SDK_SUCCESS)
    {
        Console.WriteLine("Got error({0}).", result);
    }
    else
    {
        IntPtr curResult = outResultObj;
        int resultSize =
Marshal.SizeOf(typeof(BS20sdpStandardDeviceResult));
        for (UInt32 resultIdx = ; resultIdx < numOfResult;
resultIdx++)
        {
            BS20sdpStandardDeviceResult item =
(BS20sdpStandardDeviceResult)Marshal.PtrToStructure(curResult,
typeof(BS20sdpStandardDeviceResult));
            //print(ref item, resultIdx);
            listResult.Add(item);
            curResult += resultSize;
        }

        API.BS2_ReleaseObject(outResultObj);
        Console.WriteLine("Call success.");
    }
}
finally
{
    Marshal.FreeHGlobal(ptrArray);
}
}

return result;
```

From:

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

Permanent link:

https://kb.supremainc.com/kbtest/doku.php?id=ko:bs2_updateosdpstandarddevice&rev=1675928117

Last update: **2023/02/09 16:35**