Table of Contents

BS2_SetMasterAdmin

Important
Suprema devices comply with CE RED (Radio Equipment Directive) and support the Master Admin feature.
Without admin configuration, anyone could access the admin menu and modify settings, creating a security vulnerability.
The Master Admin feature mandates full admin enrollment to prevent such risks.
Devices supporting this feature must complete master admin enrollment, and device usage is restricted until configuration is complete.
This differs from the existing operator View BS2AuthOperatorLevel structure, so please be cautious.
All device features become available after master admin enrollment is successfully completed.
Below is information about devices and versions that support the Master Admin feature.

Device Type	Supported Version
BS3	V1.4.0 and later
XS2	V1.4.0 and later
BS2a	V1.2.0 and later
BEW3	Support planned

[+ 2.9.12] Enrolls a master admin on devices supporting the CE RED (Radio Equipment Directive) Master Admin feature.

Declaration

```
#include "BS_API.h"

int BS2_SetMasterAdmin(void* context, BS2_DEVICE_ID deviceId, const
BS2UserFaceExBlob* masterAdmin);
```

View BS2UserFaceExBlob structure

Parameters

- [In] *context* : Context
- [In] *deviceId* : Device ID
- [In] *masterAdmin* : Pointer to the master admin information to enroll

Return Value

Returns BS_SDK_SUCCESS when performed successfully, and returns the corresponding error code when an error occurs.

Sample Code

C++

```
BS2UserFaceExBlob userBlob = { , };
BS2User& user = userBlob.user;

if (BS_SDK_SUCCESS != (sdkResult = getUserBlobPINCode(userBlob.pin,
deviceInfo)))
    return sdkResult;

user.numFingers = ;
user.numCards = ;
user.numFaces = ;

if (faceExScanSupported)
{
    if (BS_SDK_SUCCESS != (sdkResult =
getUserBlobFaceInfoTemplateOnly(&userBlob.faceExObjs, user.numFaces, id,
deviceInfoEx)))
        return sdkResult;
}

int sdkResult = BS2_SetMasterAdmin(context_, id, &userBlob);
if (BS_SDK_SUCCESS != sdkResult)
    TRACE("BS2_SetMasterAdmin call failed: %d", sdkResult);

if (userBlob.faceExObjs)
{
    delete[] userBlob.faceExObjs;
}

return sdkResult;
```

C#

```
BS2ErrorCode sdkResult = BS2ErrorCode.BS_SDK_SUCCESS;

BS2UserFaceExBlob userBlob = Util.AllocateStructure<BS2UserFaceExBlob>();
userBlob.user.numCards = ;
userBlob.user.numFingers = ;
userBlob.user.numFaces = ;

userBlob.cardObjs = IntPtr.Zero;
```

```
userBlob.fingerObjs = IntPtr.Zero;
userBlob.faceObjs = IntPtr.Zero;
userBlob.faceExObjs = IntPtr.Zero;

if (BS2ErrorCode.BS_SDK_SUCCESS != (sdkResult =
getUserBlobPINCode(sdkContext, ref userBlob.pin)))
    return;

if (BS2ErrorCode.BS_SDK_SUCCESS != (sdkResult =
getUserBlobFaceExInfoTemplateOnly(sdkContext, deviceID, ref
userBlob.faceExObjs, ref userBlob.user.numFaces)))
    return;

Console.WriteLine("Trying to set master admin");
sdkResult = (BS2ErrorCode)API.BS2_SetMasterAdmin(sdkContext, deviceID, ref
userBlob);
if (BS2ErrorCode.BS_SDK_SUCCESS != sdkResult)
    Console.WriteLine("BS2_SetMasterAdmin call failed {0}", sdkResult);

if (userBlob.faceExObjs != IntPtr.Zero)
{
    Marshal.FreeHGlobal(userBlob.faceExObjs);
}
```

See Also

BS2_GetMasterAdmin