

목차

시작하기	1
폴더와 파일 구성	1
프레임워크	1
워크플로우	2
호환되는 장치	3
<i>BioStar 1.x SDK와 비교</i>	3
일관성 - 독립적인 데이터 구조체와 API 제공	3
편의성 - 네트워크 인터페이스 자동 관리	5
고립성 - 스레드 세이프	5
유지보수 - 유연한 개발	5
개발 환경 구축하기	7
Visual Studio에서 새로운 프로젝트 만들기	7

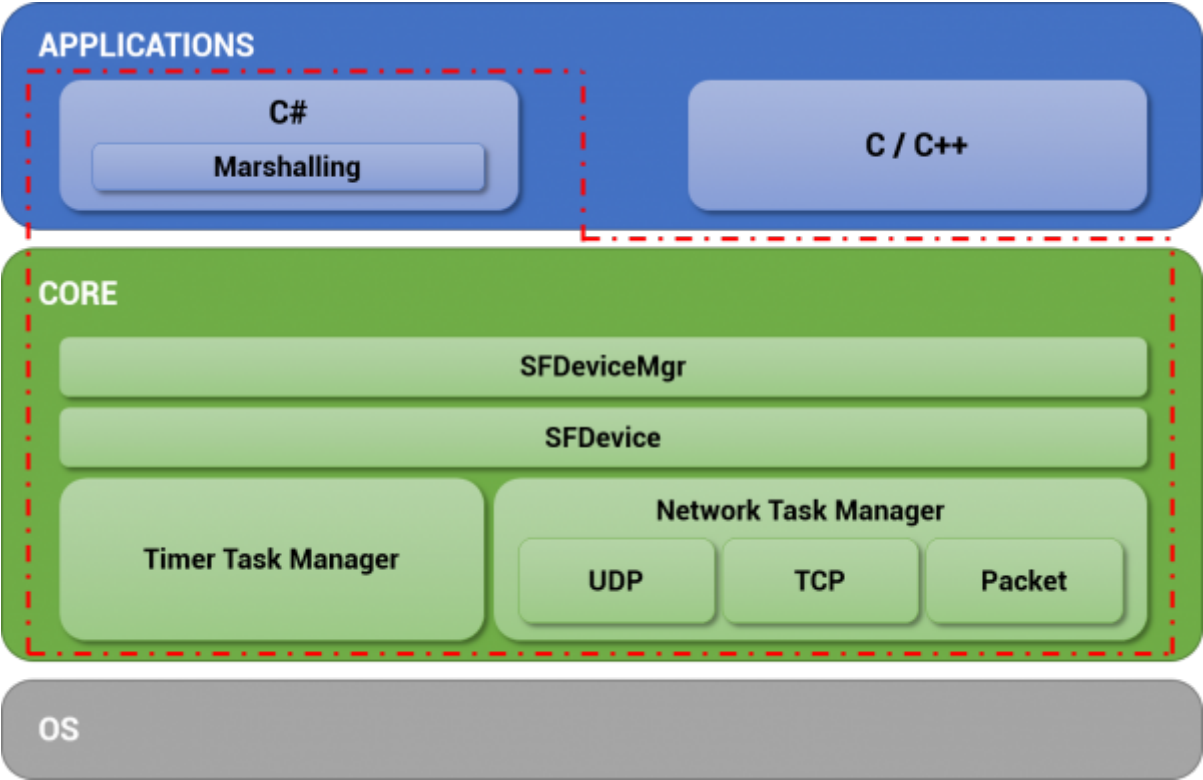
시작하기

폴더와 파일 구성

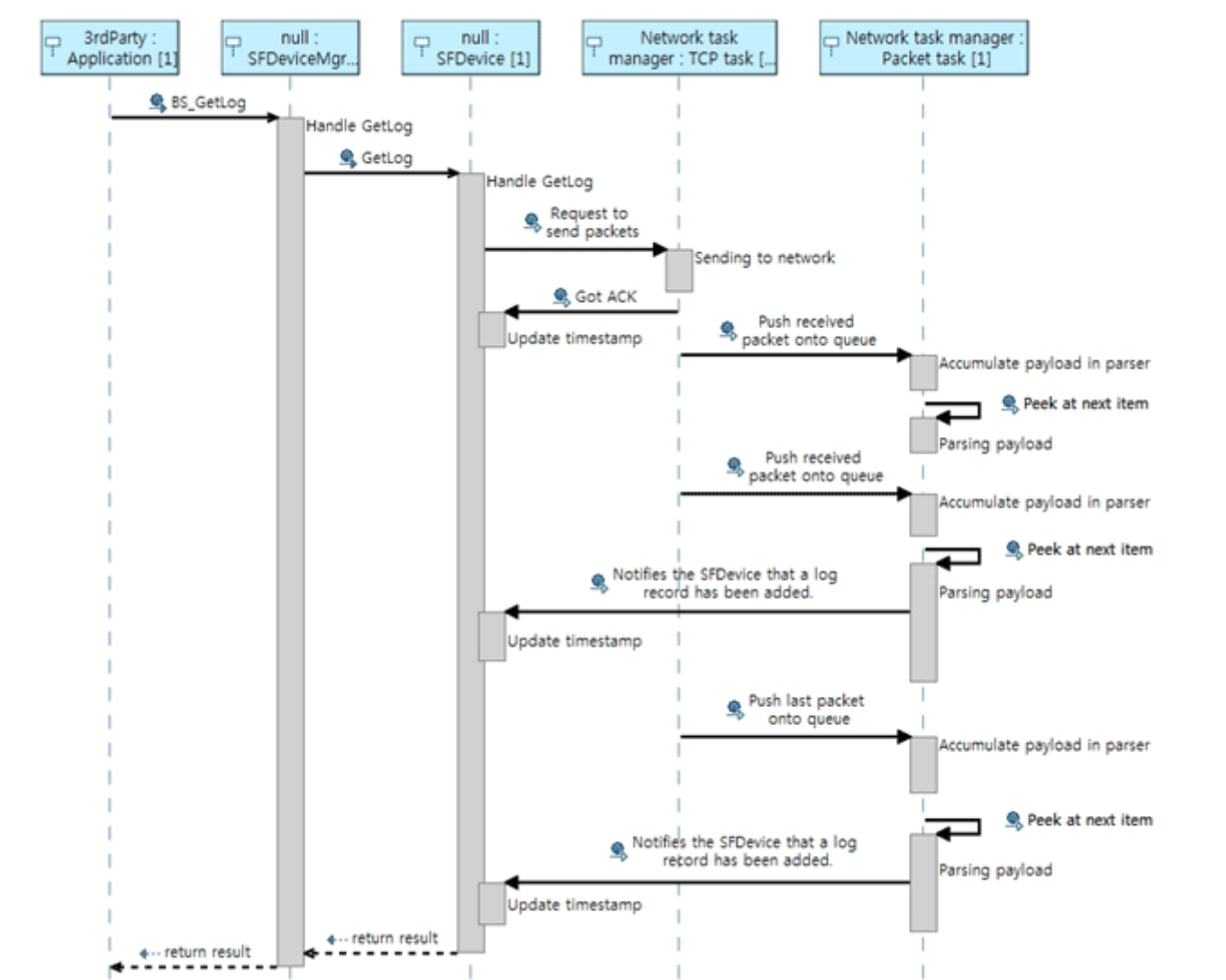
BioStar 2 Device SDK 패키지는 다음과 같은 폴더와 파일로 구성되어 있습니다.

SDK	Document ¹⁾				
	Include ²⁾				
	Lib	linux	lib	x86	BS_SDK_V2.so
				x64	BS_SDK_V2.so
		window	lib	x86	BS_SDK_V2.lib ³⁾ BS_SDK_V2.dll
				x64	BS_SDK_V2.lib ⁴⁾ BS_SDK_V2.dll
	Example ⁵⁾	C#			
		C++			

프레임워크



워크플로우



호환되는 장치

BioStar 2와 연동되는 모든 장치를 사용할 수 있습니다.

BioStar 1.x SDK와 비교

일관성 - 독립적인 데이터 구조체와 API 제공

BioStar 1.x SDK는 장치에 따라 사용할 수 있는 구조체와 API가 다릅니다. 하나의 애플리케이션에서 여러 종류의 장치를 제어하려면 장치별로 분기문을 사용해야 하므로 불편합니다.

```
if( m_DeviceType == BS_DEVICE_BIOENTRY_PLUS ||
    m_DeviceType == BS_DEVICE_BIOENTRY_W    ||
    m_DeviceType == BS_DEVICE_BIOLITE       ||
    m_DeviceType == BS_DEVICE_XPASS         ||
    m_DeviceType == BS_DEVICE_XPASS_SLIM    ||
    m_DeviceType == BS_DEVICE_XPASS_SLIM2 )
{
    BEUserHdr userHdr;
    // Retrieve a user from the device
    BS_RET_CODE result = BS_GetUserBEPlus( m_Handle, m_UserID, &userHdr,
m_TemplateData );
    ...

    // Transfer the user to the device
    result = BS_EnrollUserBEPlus( m_Handle, &userHdr, m_TemplateData );
    ...
}
else if( m_DeviceType == BS_DEVICE_BIOSTATION )
{
    BSUserHdrEx userHdr;

    BS_RET_CODE result = BS_GetUserEx( m_Handle, m_UserID, &userHdr,
m_TemplateData );
    ...

    result = BS_EnrollUserEx( m_Handle, &userHdr, m_TemplateData );
    ...
}
else if( m_DeviceType == BS_DEVICE_DSTATION )
{
    DSUserHdr userHdr;
```

```

...

    BS_RET_CODE result = BS_GetUserDStation( m_Handle, m_UserID, &userHdr,
m_TemplateData, m_FaceTemplate_DST );
    ...

    result = BS_EnrollUserDStation( m_Handle, &userHdr, m_TemplateData,
m_FaceTemplate_DST );
}
else if( m_DeviceType == BS_DEVICE_XSTATION )
{
    XSUserHdr userHdr;
    ...

    BS_RET_CODE result = BS_GetUserXStation( m_Handle, m_UserID, &userHdr);
    ...

    result = BS_EnrollUserXStation( m_Handle, &userHdr );
}
else if( m_DeviceType == BS_DEVICE_BIOSTATION2 )
{
    BS2UserHdr userHdr;
    ...

    BS_RET_CODE result = BS_GetUserBioStation2( m_Handle, m_UserID,
&userHdr, m_TemplateData );
    ...

    result = BS_EnrollUserBioStation2( m_Handle, &userHdr, m_TemplateData );
}
else if( m_DeviceType == BS_DEVICE_FSTATION )
{
    FSUserHdr userHdr;
    ...

    BS_RET_CODE result = BS_GetUserFStation( m_Handle, m_UserID, &userHdr,
faceTemplate );
    ...

    result = BS_EnrollUserFStation( m_Handle, &userHdr, m_FaceTemplate_FST
);
}

```

BioStar 2.x SDK는 장치 구분 없이 하나의 구조체와 하나의 API만을 사용합니다. 개발자는 복잡한 분기문을 사용하지 않아도 되며, 간결한 코드를 사용할 수 있습니다.

```

BS2UserBlob userBlob =
(BS2UserBlob)Utils.AllocateStructure(sizeof(BS2UserBlob));
...

```

```
int result = (BS2ErrorCode)API.BS2_EnrolUser(Program.sdkContext,
deviceHandle.info.id, ref userBlob);
...
```

편의성 - 네트워크 인터페이스 자동 관리

BioStar 1.x SDK는 장치에 접속할 때 해당 장치에 대한 핸들(소켓 기술자)을 획득해야 합니다. 획득된 핸들(소켓 기술자)을 API를 호출할 때 사용하여 어떤 장치를 제어하려는지 알립니다.

```
int handle = ;
uint deviceID = ;
int deviceType = ;

result = BS_OpenSocket( "192.168.0.5", 1471, &handle );
result = BS_GetDeviceID(handle, &deviceID, &deviceType);
```

BioStar 2.x SDK는 장치에 대한 핸들(소켓 기술자)을 개발자가 제어하지 않습니다. 장치 ID를 매개변수로 전달하면 BioStar 2.x SDK 프레임워크가 해당 장치를 자동으로 찾아서 제어합니다.

```
const char* deviceAddress = "192.168.1.2";
uint16_t devicePort = 51211;
uint32_t deviceId = ;
BS2SimpleDeviceInfo deviceInfo;

int result = BS2_ConnectDeviceViaIP(context, deviceAddress, devicePort,
&deviceId);
int result = BS2_GetDeviceInfo(context, deviceId, &deviceInfo);
```

고립성 - 스레드 세이프

BioStar 1.x SDK는 하나의 API가 여러 스레드에서 동시에 호출되지 않도록 개발자가 직접 '락 매커니즘'을 구성해야 합니다.

BioStar 2.x SDK는 사용 중인 API를 다른 스레드에서 동시에 호출할 수 있도록 설계되었습니다.

유지보수 - 유연한 개발

BioStar 1.x SDK는 신규 장치가 추가되면 애플리케이션의 UI/로직을 추가하거나 수정해야 합니다. 하지만, **BioStar 2.x SDK**는 공통된 구조체로 각 장치의 특성 정보를 제공하므로 신규 장치가 추가되더라도 기존 애플리케이션의 UI/로직을 수정할 필요가 없습니다.

예를 들어, 얼굴 인증 정보를 지원하는 신규 장치가 출시되더라도 애플리케이션의 UI를 설계할 때 각 장

치의 특성 정보에 따라 UI/로직이 동작하도록 설계했다면 신규 장치가 추가되어도 애플리케이션을 수정하는 번거로움을 덜 수 있습니다.

개발 환경 구축하기

Visual Studio에서 새로운 프로젝트 만들기

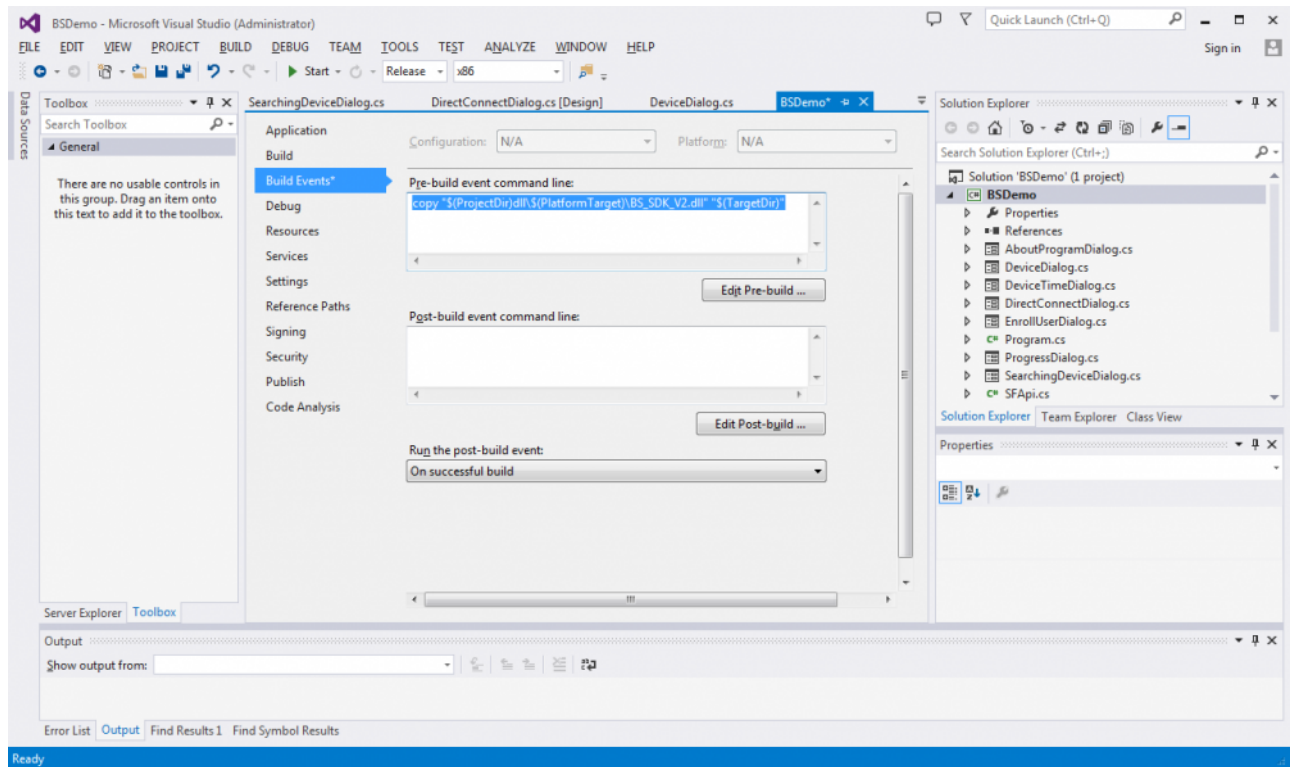
C/C++
Under construction

C#

- 1. SDK 패키지에서 라이브러리 디렉토리를 선택하고 프로젝트 디렉토리로 복사하십시오.
- 2. 플랫폼 대상에 맞는 올바른 DLL를 사용하기 위해 프로젝트 속성을 수정해야 합니다.
프로젝트 속성 페이지를 열고 빌드 전 이벤트 명령줄에 다음과 같이 입력하십시오.

SDK 버전	플랫폼	입력 정보
V2.8.1 이전	-	copy "\$(ProjectDir)lib\\$(PlatformTarget)\BS_SDK_V2.dll" "\$ (TargetDir)" copy "\$(ProjectDir)lib\\$(PlatformTarget)\libeay32.dll" "\$ (TargetDir)" copy "\$(ProjectDir)lib\\$(PlatformTarget)\libssl32.dll" "\$ (TargetDir)" copy "\$(ProjectDir)lib\\$(PlatformTarget)\ssleay32.dll" "\$ (TargetDir)"
V2.8.2 이후	x86	copy "\$(ProjectDir)lib\\$(PlatformTarget)\BS_SDK_V2.dll" "\$ (TargetDir)" copy "\$(ProjectDir)lib\\$(PlatformTarget)\libssl-1_1.dll" "\$ (TargetDir)" copy "\$(ProjectDir)lib\\$(PlatformTarget)\libcrypto-1_1.dll" "\$ (TargetDir)"
V2.8.2 이후	x64	copy "\$(ProjectDir)lib\\$(PlatformTarget)\BS_SDK_V2.dll" "\$ (TargetDir)" copy "\$(ProjectDir)lib\\$(PlatformTarget)\libssl-1_1-x64.dll" "\$ (TargetDir)" copy "\$(ProjectDir)lib\\$(PlatformTarget)\libcrypto-1_1-x64.dll" "\$ (TargetDir)"

- 1. SDK 패키지의 예제 코드에서 **SFApi.cs**, **SFEnum.cs**, **SFStruct.cs**를 복사하십시오.



- 1) SDK에서 제공하는 API의 사용법을 기술한 문서 파일이 존재합니다.
- 2) API 및 구조체를 정의한 헤더 파일들이 있으며, C/C++ 애플리케이션 개발 시 필요합니다.
- 3) , 4) C/C++ 프로젝트에서 import할 정적 라이브러리입니다.
- 5) 다양한 언어별로 SDK 샘플 코드가 존재합니다.

From:

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

Permanent link:

https://kb.supremainc.com/kbtest/doku.php?id=ko:getting_started&rev=1630555386

Last update: **2021/09/02 13:03**