

# Intel® RealSense™ SDK 2014

## Getting Started

With the Intel® RealSense™ SDK, you have access to robust, natural human-computer interaction (HCI) algorithms such as face tracking, finger tracking, gesture recognition, voice and speech recognition, fully textured 3D scanning and enhanced depth augmented reality.

With the SDK you can create Windows\* desktop applications that offer innovative user experiences.

After performing the steps in this tutorial, you'll be ready to work through the [Capturing Raw Stream Tutorial](#) and start using the SDK.

# Contents

- **Quick Start**
- **The Intel® RealSense™ SDK Architecture**
- **Hardware and Software Requirements and Tools**
- **Install the SDK and Set up the Camera**
  - Install the SDK
  - Set up the Camera
- **Set up Your C++ Development Environment**
- **Run the Hello Intel RealSense SDK App**
  - Create a Session
  - Retrieve a Created Session
  - Run the App
- **To learn more**

# Quick Start

To get started quickly, follow these steps:

1	2	3
<a href="#">Download and install the SDK</a>	<a href="#">Set up your development environment</a> and verify setup by running <a href="#">Hello Intel RealSense SDK</a>	Read the next tutorial, <a href="#">Capturing Raw Streams</a> .

To learn more about the SDK, read the other sections in this document.

# The Intel® RealSense™ SDK Architecture

As you can see in Figure 1, applications that integrate the SDK sit on three layers of components if the wrappers are used; two layers if the SDK interfaces are accessed directly.

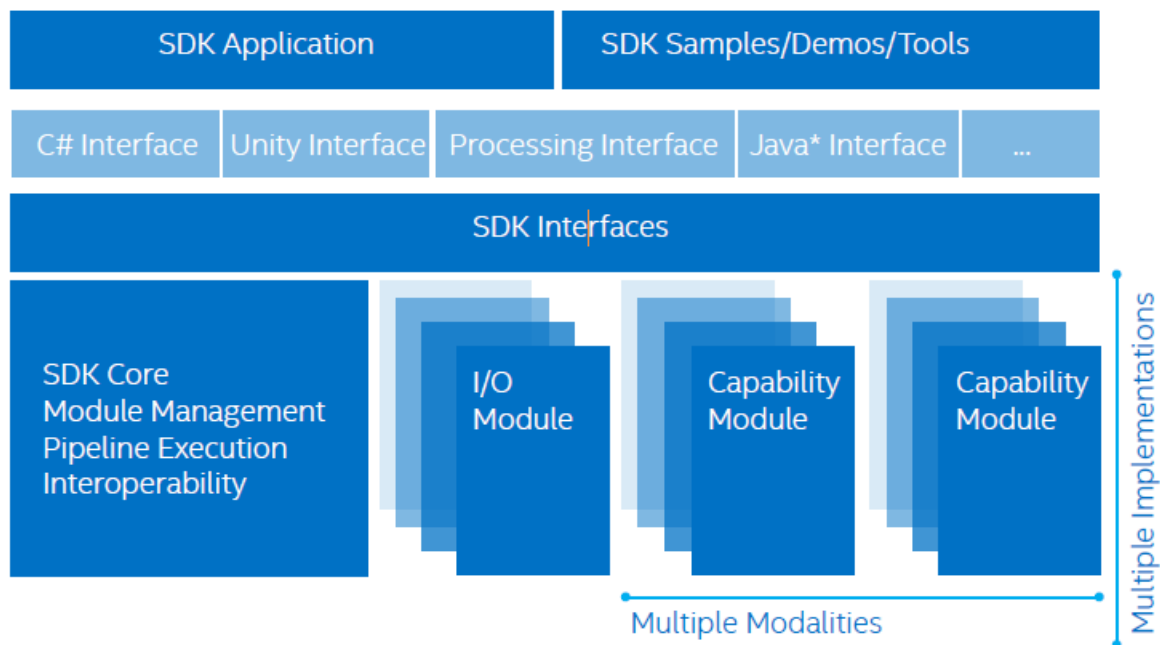


Figure 1. The Intel® RealSense™ SDK Architecture

The wrappers expose the SDK function calls in a variety of languages. The SDK interfaces are the C++ exposed API function calls.

The core is responsible for organizing the execution pipeline. It is also the base of the components that manages the two types of modules that provide SDK functionalities to your application.

- **I/O modules:** Captures the input data from your device and sends that data to an output device or to the algorithm modules.
- **Algorithm modules:** Includes various pattern detection and recognition algorithms, like face tracking and recognition, finger tracking, gesture recognition, and voice recognition and synthesis.

It is possible to have multiple modules contained within the pipeline at the same time, so it is essential that the pipeline have a manager. If you want to utilize more than one camera or other input device in your application, you may require multiple pipelines, each with its own manager.

# Hardware and Software Requirements and Tools

Required Hardware	4th generation (code name Haswell) Intel® Core™ processor 2GB free hard disk space Intel® RealSense™ 3D Camera USB 3.0 Port
Required OS	Microsoft Windows* 8 or 8.1 OS 64-bit
Supported Languages	C++, C#
Supported IDE	Microsoft Visual Studio* C++ 2010-2013 with service pack 1 or newer Unity* PRO 4. 1. 0 or later for Unity game development
Optional Programming Languages	Java* (JDK 1.7.0_11 or later) Processing* 2.1.2 or later

# Install the SDK and Set up the Camera

## Install the SDK

1. Download and run the SDK installer from <https://software.intel.com/realsense>
2. You will see a welcome screen as illustrated in Figure 2. Follow the instructions to complete the installation process.
  - By default, the SDK installs to the `C:/ProgramFiles(x86)/Intel/RSSDK` directory.
  - If the SDK installer detects any existing SDK versions, the SDK installer will prompt you for an upgrade. It is recommended to always do a clean uninstall and then install any newer SDK versions.

After installation, reboot the system when prompted. (This step is critical to propagate all environmental variables.)



Figure 2. Installer Welcome Screen

## Set up the Camera

1. If the camera is integrated into the computer or laptop, skip to step 4.
  2. Install the camera on top of the computer or laptop lid.
  3. Plug the USB connection into one of the USB ports, as shown in Figure 3.
  4. Position yourself comfortably, with your back supported by your chair in a relaxed position, so your hands can move freely in front of the camera.
- To avoid fatigue, it is critical that users be in a relaxed position.

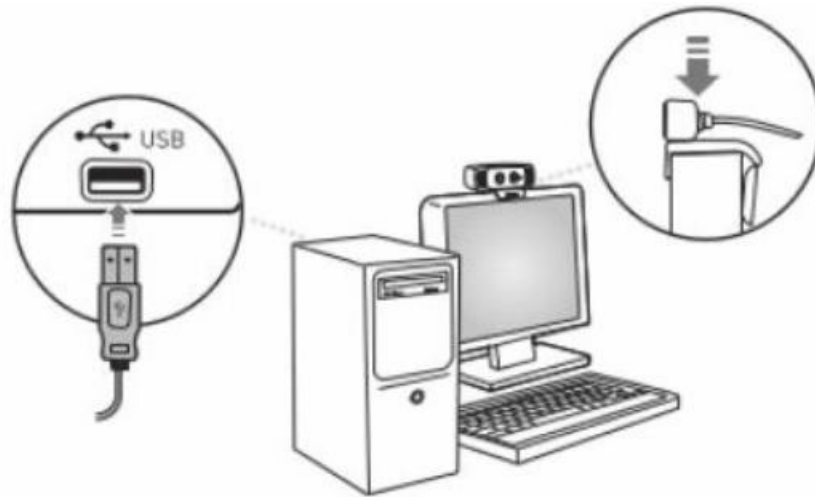


Figure 3. Camera Setup

# Set up Your C++ Development Environment

The best way to set up your environment is by importing the SDK's property sheets.

1. Create a new project or open an existing project in **Visual Studio\***.
2. Open the property manager using **View->Other Windows->Property Manager**.
3. Right-click the project name and then click **Add Existing Property Sheet**.
4. Locate the SDK integration property sheets found in the directory:  
[C:\Program Files \(x86\)\Intel\RSSDK\props](C:\Program Files (x86)\Intel\RSSDK\props)
5. Choose the appropriate property sheet depending on whether your application requires dynamic or static runtime:

## Property Sheets and Descriptions

<b>Compile option</b>	<b>Microsoft Visual Studio* 2010-2013</b>
dynamic runtime	VS2010-13.Integration.MD.props
static runtime	VS2010-13.Integration.MT.props

Now, go to the next section and run the “Hello Intel RealSense SDK” app to be sure you have properly set up your development environment.



# Run the Hello Intel RealSense SDK App

Running this application will help you make sure your development environment is set up properly.

## Create a Session

The SDK core is represented by two interfaces:

- **PXCSession** manages all of the modules of the SDK
- **PXCSenseManager** organizes a pipeline by starting, stopping, and pausing the operations of its various modalities.

Each session maintains its own pipeline that contains the I/O and algorithm modules.

1. Create an instance of **PXCSenseManager**
2. **CreateInstance()** of the **PXCSenseManager** class creates both an instance of **PXCSenseManager** and an underlying instance of **PXCSession**.

```
// create the PXCSenseManager
PXCSenseManager *psm=0;
psm = PXCSenseManager::CreateInstance();

if (!psm) {
    wprintf_s(L"Unable to create the PXCSenseManager\n");
    return 1;
}
```

## Retrieve a Created Session

Often you will need to retrieve one or more of the sessions you created. Session retrieval allows you to manage input device operations such as setting device properties and enumerating input streams. You can also use the session instance to enumerate the I/O and algorithm modules that are available in the SDK.

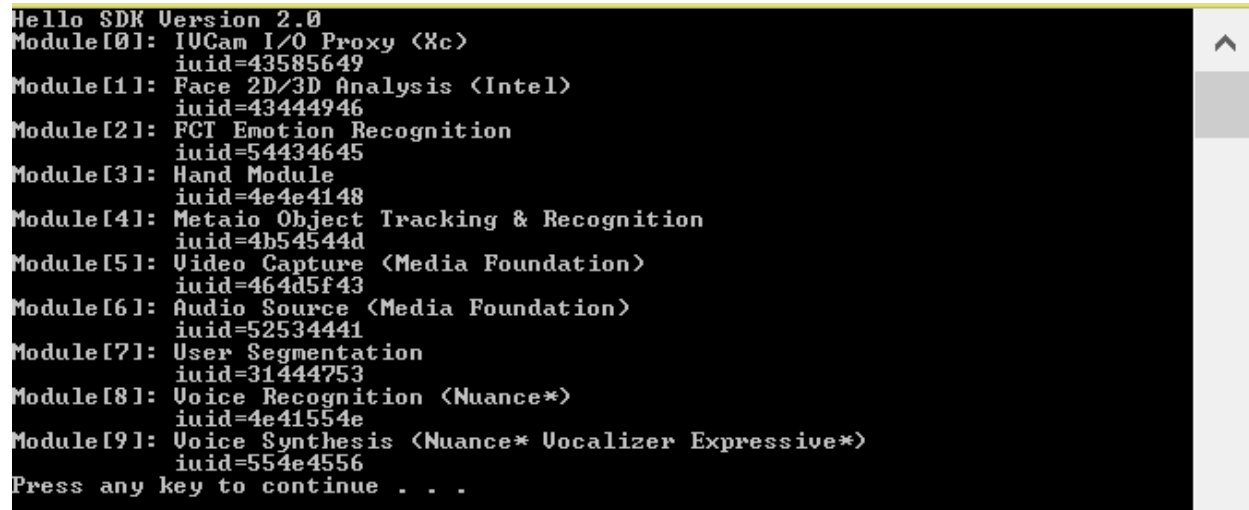
1. Create an instance of **PXCSession**
2. **QuerySession()** of the PXCSessionManager class returns a session of a particular **PXCSessionManager** instance.

```
PXCSession *session;  
  
session = psm->QuerySession();
```

## Run the App

You can create a new C++ file and copy-paste the code below. You can also build and run the existing **1\_GettingStarted** sample in Visual Studio, or run the executables found in the “Debug” subfolder of the code sample directory.

A command prompt window opens with the names and ids of all the I/O and algorithm modules that were loaded when you installed the SDK (Figure 4).



```
Hello SDK Version 2.0  
Module[0]: IUCam I/O Proxy (Xc)  
            iuid=43585649  
Module[1]: Face 2D/3D Analysis (Intel)  
            iuid=43444946  
Module[2]: FCT Emotion Recognition  
            iuid=54434645  
Module[3]: Hand Module  
            iuid=4e4e4148  
Module[4]: Metaio Object Tracking & Recognition  
            iuid=4b54544d  
Module[5]: Video Capture (Media Foundation)  
            iuid=464d5f43  
Module[6]: Audio Source (Media Foundation)  
            iuid=52534441  
Module[7]: User Segmentation  
            iuid=31444753  
Module[8]: Voice Recognition (Nuance*)  
            iuid=4e41554e  
Module[9]: Voice Synthesis (Nuance* Vocalizer Expressive*)  
            iuid=554e4556  
Press any key to continue . . .
```

Figure 4. Result of running Hello Intel RealSense SDK App

```

#include <windows.h>
#include <wchar.h>
#include "pxcsensemanager.h"

int wmain(int argc, WCHAR* argv[]) {

    // create the PXCSessionManager
    PXCSessionManager *psm=0;
    psm = PXCSessionManager::CreateInstance();
    if (!psm) {
        wprintf_s(L"Unable to create the PXCSessionManager\n");
        return 1;
    }

    // Retrieve the underlying session created by the PXCSessionManager.
    // The returned instance is an PXCSessionManager internally managed object.
    // Note: Do not release the session!
    PXCSession *session;
    session = psm->QuerySession();
    if (session == NULL) {
        wprintf_s(L"Session not created by PXCSessionManager\n");
        return 2;
    }

    // query the session version
    PXCSession::ImplVersion ver;
    ver = session->QueryVersion();

    // print version to console
    wprintf_s(L" Hello Intel RSSDK Version %d.%d \n",ver.major, ver.minor);

    // enumerate all available modules that are automatically loaded with the RSSDK
    for (int i=0;i++) {
        PXCSession::ImplDesc desc;
        if ( session->QueryImpl(0,i,&desc) < PXC_STATUS_NO_ERROR ) break;

        // Print the module friendly name and iuid (interface unique ID)
        wprintf_s(L"Module[%d]: %s\n",i,desc.friendlyName);
        wprintf_s(L"      iuid=%x\n",desc.iuid);
    }

    // close the streams (if any) and release any session and processing module instances
    psm->Release();
    system("pause");
    return 0;
}

```

## To learn more

- For more information, read the [Architecture](#) and the [Programming Guide](#) sections in the SDK Reference Manual.