



# **Intel<sup>®</sup> RealSense<sup>™</sup> SDK Developer Guide**

# Table of Contents

<b>Legal Information</b>	<b>4</b>
<b>Introducing the SDK</b>	<b>6</b>
<b>Getting Started with the SDK</b>	<b>7</b>
<b>Installing the SDK</b>	<b>8</b>
<b>Setting Up the Camera</b>	<b>9</b>
<b>Configuring Application Development Environment</b>	<b>11</b>
Configuring C++ Development Environment.....	12
Importing the SDK Property Sheets .....	13
Configuring Project Settings .....	14
Configuring C# Development Environment.....	16
Customize the C# libraries .....	17
Configuring Unity Application Development Environment.....	18
Customizing the Unity Libraries .....	19
Configuring Processing Application Development Environment.....	20
Configuring Java Application Development Environment.....	21
Configuring JavaScript Application Development Environment.....	22
<b>Application Deployment Guidelines</b>	<b>23</b>
Runtime Installers .....	24
Command Option and Status Code .....	27
Installation Features .....	29
Supporting Libraries .....	31
Checking Module Availability.....	32
Checking Camera Driver Version.....	33
Privacy Requirements and Guidelines.....	34
Example of Data Type Description .....	35
Example of Privacy Notification Page .....	36
<b>Working with Multiple Modalities</b>	<b>37</b>
<b>Coexisting with Other Applications</b>	<b>44</b>
Sharing Device Configuration.....	45
Sharing Device Properties.....	51

---

Sharing Platform Computations.....	53
<b>Managing Power and Performance</b>	<b>54</b>
<b>Algorithm Operating Ranges</b>	<b>56</b>

# 1 Legal Information

By using this document, in addition to any agreements you have with Intel, you accept the terms set forth below.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

Information in this document is provided in connection with Intel products. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted by this document. Except as provided in Intel's Terms and Conditions of Sale for such products, Intel assumes no liability whatsoever and Intel disclaims any express or implied warranty, relating to sale and/or use of Intel products including liability or warranties relating to fitness for a particular purpose, merchantability, or infringement of any patent, copyright or other intellectual property right.

Unless otherwise agreed in writing by Intel, the Intel products are not designed nor intended for any application in which the failure of the Intel product could create a situation where personal injury or death may occur.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "Reserved" or "Undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or by visiting Intel's web site <http://www.intel.com>.

Any software source code reprinted in this document is furnished under a software license and may only be used or copied in accordance with the terms of that license.

Intel, the Intel logo, Intel RealSense, Intel Core and Iris are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Microsoft, Windows, and the Windows logo are trademarks, or registered trademarks of Microsoft Corporation in the United States and/or other countries.

Java is a registered trademark of Oracle and/or its affiliates.

\*Other names and brands may be claimed as the property of others.

Copyright © 2010-2014, Intel Corporation. All rights reserved.

## 2 Introducing the SDK

The Intel® RealSense™ SDK is a library of pattern detection and recognition algorithm implementations exposed through standardized interfaces. The library aims at lowering barriers to using these algorithms and shifting the application developers' focus from coding the algorithm details to innovating on the usage of these algorithms for next generation human computer experience.

### Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor- dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

**Notice revision #20110804**

This document addresses application development setup and product packaging details.

### Notational Conventions

This SDK document uses the Calibri typeface for normal prose.

With the exception of section headings, captions and the table of contents, all code-related items appear in the Courier New typeface (`pxcStatus`).

Hyperlinks appear underlined in blue, such as [pxcStatus](#).



This is a note that provides additional information to aid your understanding of the procedure or concept.



This is a tip that provides alternate methods or shortcuts.



This is a result statement which indicates what you can expect to see or happen after performing a step.

### 3 Getting Started with the SDK

The SDK requires the following hardware and software setup:

#### Hardware requirements:

- 4th generation Intel® Core™ processors based on the Intel microarchitecture code name Haswell
- 8GB free hard disk space
- Intel® RealSense™ 3D camera (required to connect to a USB\* 3 port)


#### Software requirements:


- Microsoft\* Windows\* 8.1 OS 64-bit
- Microsoft Visual Studio\* 2010-2013 with the latest service pack
- Microsoft .NET\* 4.0 Framework for C# development
- Unity\* PRO 4.1.0 or higher for Unity game development
- Any of the following browsers for JavaScript development:
  - Microsoft\* Internet Explorer\* 10.0.13
  - Google\* Chrome\* 33.0.1750.146
  - Mozilla\* Firefox\* 27.0.1
- Processing\* 2.1.2 or higher for Processing development
- Java\* JDK 1.7.0\_11 or higher for Java development

## 4 Installing the SDK

To install the SDK software, complete the following steps:

1. Download and run the SDK installer from <http://www.intel.com/software/perceptual>.
2. You will see a welcome screen as illustrated in Figure 1. Follow the installer instructions to complete the installation process.

 By default, the SDK installs to the `C:/Program Files (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.

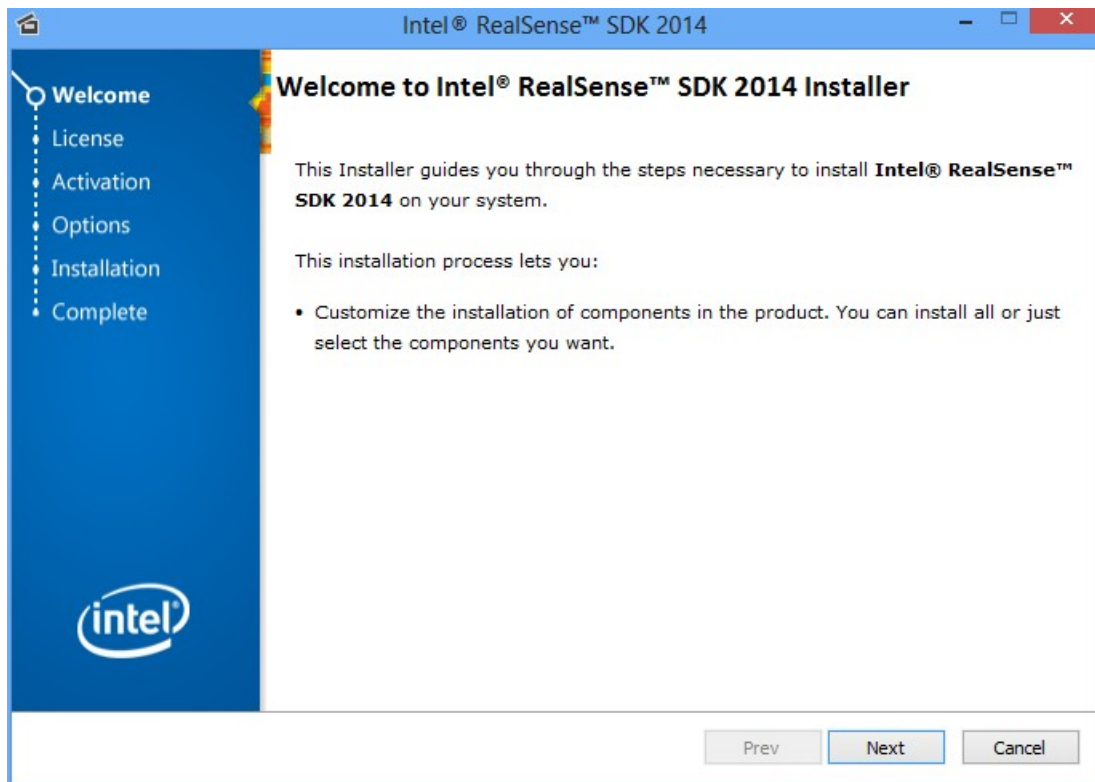


Figure 1: Installer Welcome Screen

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



## 5 Setting Up the Camera

To setup the camera, complete the following steps. If the camera is already integrated into the computer or laptop, skip to step 3.

1. Install the camera on top of the computer or laptop lid.
2. Plug the USB connector into one of the USB3 ports, as illustrated in Figure 1.

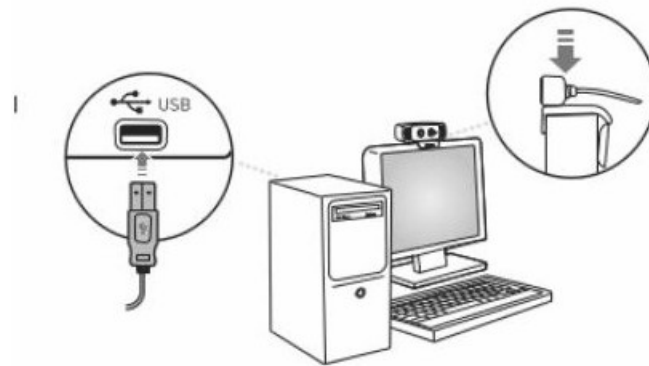


Figure 1: Camera Setup Position

3. Position yourself comfortably, with your back supported by the chair in a relaxed position, so that your hands can move freely in front of the camera.



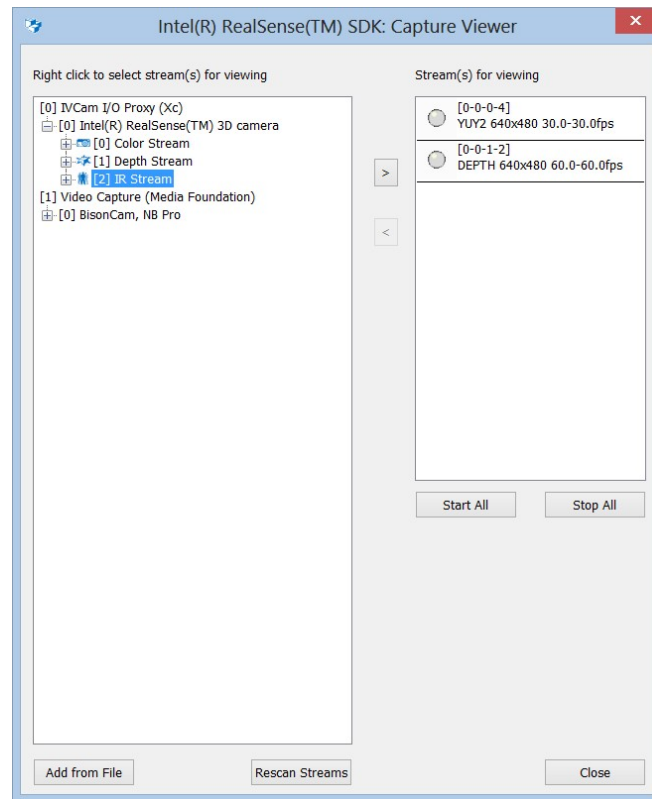
To avoid fatigue, it is critical to find a relaxed position that satisfies the above requirement.

Here are the steps to check if the SDK and the camera are installed correctly:

1. From the `Startup` menu, select `Startup>Intel® RealSense™ SDK>Tools>Capture Viewer` to launch the `capture_viewer` application.
2. Select a color stream and a depth stream.
3. Click `Start All` to render the streams.



If your screen looks similar to the one shown in Figure 2, the camera is working as expected.



**Figure 2: Camera Testing for Color and Depth**

## 6 Configuring Application Development Environment

This section describes how to configure the development environment for C++, C# and Unity applications.

## 6.1 Configuring C++ Development Environment

The SDK provides two ways to setup the C++ development environment:

- For easy integration, the application can use the integration [Property Sheets](#)<sup>13</sup>.
- For addition flexibility, the application can directly set the [Project Settings](#)<sup>14</sup>.

## 6.1.1 Importing the SDK Property Sheets

For easy integration into the Microsoft Visual Studio development environment, use the property sheets located under the `$(RSSDK_DIR)/props` directory. Table 1 lists available property sheets.

Property Sheet	Description
<code>VS2010-13.Integration.MD.props</code>	Microsoft Visual Studio 2010-2013 property sheet for applications that compile with the dynamic runtime option.
<code>VS2010-13.Integration.MT.props</code>	Microsoft Visual Studio 2010-2013 property sheet for applications that compile with the static runtime option.

**Table 1: Visual Studio Integration Property Sheets**

To import the property sheets, complete the following steps:

- Create a new project or open an existing project.
- Open the property manager by `View->Property Manager`.
- Right click on the project name and choose `Add Existing Property Sheet`. Add `VS2010-13.Integration.MD.props` or `VS2010-13.Integration.MT.props` for the application that requires dynamic or static runtime, respectively.

## 6.1.2 Configuring Project Settings

Alternatively, you can directly modify the development settings as follows:

- Include Paths:

Add any required include paths from Table 2 to the respective project setting.

Include Path	Description
<code>\$(RSSDK_DIR)/include</code>	Required to access I/O module and algorithm module functions
<code>\$(RSSDK_DIR)/sample/common/include</code>	Required to access any utility class functions

Table 2: Include Paths

- Library Files and Library Paths:

For applications that compile with the static runtime option, the SDK provides prebuilt libraries listed in Table 3. Add any required library paths and library files to the respective project setting.

Library Path	Library File	Prebuilt Library Description
<code>\$(RSSDK_DIR)/lib/\$(PlatformName)</code>	<code>libpxc.lib (RELEASE)</code> <code>libpxc_d.lib (DEBUG)</code>	SDK library with the static runtime option
<code>\$(RSSDK_DIR)/sample/common/lib/\$(PlatformName)/\$(PlatformToolset)</code>	<code>libpxcutils.lib (RELEASE)</code> <code>libpxcutils_d.lib (DEBUG)</code>	SDK utility library with the static runtime option

Table 3: Library Paths and Library Files

For applications that compile with the dynamic runtime option, the SDK provides prebuilt libraries listed in Table 4. Add any required library paths and library files to the respective project setting.

Library Path	Library File	Prebuilt Library Description
<code>\$(RSSDK_DIR)/lib/\$(PlatformName)</code>	<code>libpxcmd.lib (RELEASE)</code> <code>libpxcmd_d.lib (DEBUG)</code>	SDK library with the dynamic runtime option

<code>\$(RSSDK_DIR)/sample/</code>	<code>libpxcutilsmd.lib</code>	SDK utility library with the dynamic runtime option
<code>common/</code>	<code>(RELEASE)</code>	
<code>lib/\$(PlatformName)/\$(Pla</code>	<code>libpxcutilsmd_d.lib</code>	
<code>tformToolset)</code>	<code>(DEBUG)</code>	

**Table 4: Library Paths and Library Files**

For any special compilation options that are not covered by the prebuilt libraries, you can re-compile library source files that are listed in Table 5.

Source Files	Description
<code>\$(RSSDK_DIR)/src/libpxc/*.cpp</code>	SDK library source files
<code>\$(RSSDK_DIR)/sample/common/src/*.cpp</code>	SDK utility library source files

**Table 5: Library and Utility Library Source Files**

## 6.2 Configuring C# Development Environment

The SDK exposes the C# interfaces as two dynamically linked libraries (DLL), which support Microsoft .NET framework 4.0.

- **C# interface DLL:** `libpxcclr.cs.dll`
- **C++ P/Invoke DLL:** `libpxccpp2c.dll`

Use the following steps to add DLL as a reference in Microsoft Visual Studio 2010-2012:

- Create a new project or open an existing project.
- In the solution explorer, right click the project name and select `Add Reference`.
- Add `$(RSSDK_DIR)/bin/win32/libpxcclr.cs.dll` or `$(RSSDK_DIR)/bin/x64/libpxcclr.cs.dll` as the reference.



It is a known limitation that Microsoft Visual Studio cannot handle 32-bit and 64-bit references at the same time. Thus the application must explicitly modify the reference before building a different target. See this [forum post](#) for a possible workaround.



If the application uses `Copy Local=True`, the application must manually copy `libpxccpp2c.dll` to your local directory.



## 6.2.1 Customize the C# libraries

You can customize the C# wrapper libraries to add any missing features, such as certain format conversion functions for C# applications.

The source code of the `libpxccclr.cs.dll` is under `$(RSSDK_DIR)/framework/common/pxccclr.cs`.

- Click `libpxccclr.cs_vs2010-13.sln` to launch the solution.
- Modify the code to add any missing features.
- Build and save the rebuilt library `libpxccclr.cs.dll`, under the local `bin` directory.

The source code of the library `libpxccpp2c.dll` is under `$(RSSDK_DIR)/framework/common/pxccpp2c`.

- Click the corresponding solution file, for example, `libpxccpp2c_vs2012.sln` to launch the solution.
- Modify the code to add any missing features.
- Build and save the library `libpxccpp2c.dll`, under the local `bin` directory.

## 6.3 Configuring Unity Application Development Environment

Complete the following steps to configure the environment for Unity game application development:

### PC Standalone Application

- Create a new Unity project.
- Import the Unity package `$(RSSDK_DIR)/framework/Unity/Unity.PCStandalone.unitypackage`, or do it manually as follows:
  - Create a `Plugins` directory under `Assets`.
  - Copy the following file(s) from `$(RSSDK_DIR)/bin/win32` or `$(RSSDK_DIR)/bin/x64`:
    - `libpxccpp2c.dll`: This file is the `P/Invoke` library of the SDK functions.
  - Create a `Plugins.Managed` directory under `Assets`.
  - Copy the following file(s) from `$(RSSDK_DIR)/bin/win32` or `$(RSSDK_DIR)/bin/x64` and `$(RSSDK_DIR)/framework/common/pxcclr.cs/src`:
    - `libpxcclr.unity.dll`: This file is the Unity-wrapper of the SDK functions.
    - `pxcmdefs.extensions.cs`: This optional file provides additional type extension functions.

### Webplayer Application

- Create a new Unity project.
- Import the Unity package `$(RSSDK_DIR)/framework/Unity/Unity.Webplayer.unitypackage`, or do it manually as follows:
  - Create a `Plugins.Managed` directory under `Assets`.
  - Copy the following files from `$(RSSDK_DIR)/bin/win32` and `$(RSSDK_DIR)/framework/common/pxcclr.cs/src`:
    - `libpxcclr.unityweb.dll`: This file is the Unity-wrapper of the SDK functions.
    - `pxcmdefs.extensions.cs`: This optional file provides additional type extension functions.
    - `WebSocket4Net.dll`: This file provides client side web socket support.

See [Browser Support](#) for additional details how to setup the web player start page.

### 6.3.1 Customizing the Unity Libraries

You can customize the supporting libraries for Unity to add any missing features, such as certain format conversion functions for Unity applications.

The source code of the library `libpxccclr.unity.dll` is under `$(RSSDK_DIR)/framework/common/pxccclr.cs`.

- Click `libpxccclr.unity_vs2010-13.sln` to launch the solution.
- Modify the code to add any missing features.
- Build and save the rebuilt library `libpxccclr.unity.dll`, under the local `bin` directory.

The source code of the library `libpxccpp2c.dll` is under `$(RSSDK_DIR)/framework/common/pxccpp2c`.

- Click the corresponding solution file, for example, `libpxccpp2c_vs2012.sln` to launch the solution.
- Modify the code to add any missing features.
- Build and save the library `libpxccpp2c.dll`, under the local `bin` directory.

## 6.4 Configuring Processing Application Development Environment

Do the following steps to setup the environment for Processing application development:

- Copy everything under `$(RSSDK_DIR)/framework/Processing/libraries` to the Processing sketch directory.

💡 In the Processing sketch, import the SDK name space by `"import intel.rssdk.*;"`

## 6.5 Configuring Java Application Development Environment

To setup the Java environment, copy all files under `$(RSSDK_DIR)/framework/common/pxcclr.java/bin/x64/*` to the Java application directory.

💡 In the Java application, import the SDK name space by `"import intel.rssdk.*;"`

To build and run the Java application, use the following scripts:

```
javac -classpath libpxcclr.java.jar *.java
java -classpath libpxcclr.java.jar;. <java-class>
```

where `<java-class>` is the main Java class of the application.

## 6.6 Configuring JavaScript Application Development Environment

No special setup is needed for JavaScript application development.

In your JavaScript application, reference the following libraries:

- **JQuery:** You can download JQuery from many browser hosting sites, for example, <https://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js>.
- **Promise:** If not natively supported by your browser, you can download Promise from <https://www.promisejs.org/polyfills/promise-done-6.0.0.min.js>.
- `$(RSSDK_DIR)/framework/common/JavaScript/realSense-4.0.js`
- `$(RSSDK_DIR)/framework/common/JavaScript/realSenseInfo-4.0.js`

See Browser Support for additional details on how to program JavaScript.

## **7      Application Deployment Guidelines**

This section describes the application deployment details.

## 7.1 Runtime Installers

The SDK includes runtime installers under `$(RSSDK_DIR)/runtime`. The directory exists if you select redistributable components, selected by default, during the SDK developer package installation.

The runtime installers perform the following operations:

- If there is no existing runtime, install the SDK modules.

Certain SDK module runtimes may have license restrictions. See [Speech Runtime and Language Packs](#) <sup>26</sup> for details.

- If there is an existing runtime, update the runtime.

The upgrading policy is illustrated in Figure 3. If the existing runtime is older (the major version is the same and the minor version is lower), the existing runtime is updated. Otherwise the two runtimes coexist on the same system. The runtime installers do not override SDK runtimes that are up-to-date, or of a different major version.

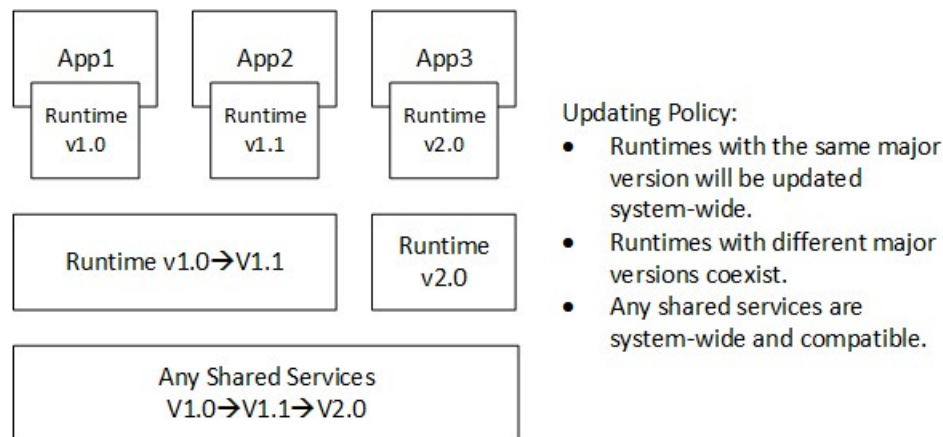


Figure 3: Runtime Upgrading Policies

Table 6 lists available runtime installers.

### Runtime Installer (YYYY is the SDK Description version)

<code>intel_rs_sdk_runtime_YYYY.exe</code>	The SDK runtime master installer for offline installation. This installer may be available only
--	---





through download due to its size.

`intel_rs_sdk_runtime_websetup_YYYY.exe` The SDK runtime master installer for online installation.

`intel_rs_sdk_runtime_core_YYY.Y.exe` The SDK core runtime installer.

 **The runtime installers may have different distribution rights. See `redist.txt` in the SDK package for details.**

 The installation process may take some time. Your application installer must wait for the installation process to complete.

 The installer requires the elevated privilege to install the runtime libraries and data files into the system folders.

You can pack the runtime installer(s) in your application installer using any of the following installation scenarios.

### Installation Case 1: Capture Only Installation

The SDK core runtime installer includes libraries for raw camera data capturing and SDK essential features such as session management, SenseManager pipelining, and file recording and playback. If your application works with camera data capturing only and does not use any algorithms (such as face tracking or hand tracking), you can pack the core runtime installer as part of the application installation.

### Installation Case 2: Module Offline Installation

The SDK runtime master installer includes the core runtime and additionally all algorithm (module) runtimes for offline installation. Usually an application only uses a subset of the provided algorithms. The master installer can export a set of customized installation scripts that include only the algorithms needed.

To generate the customized installation scripts, perform the following steps:

- Run the SDK runtime master installer with the command line `--pre-bundle=<dir>`, where `<dir>` is the absolute path of where the installer script should be exported.
- Go through the installation flow and select the modules that your application needs. No installation is done. Instead, the master installer exports the installation scripts to the destination directory.
- Pack the entire destination directory into your application installer, and follow the commands in `readme_cmd.txt` to install the SDK runtimes.

### Installation Case 3: Module Online Installation

To minimize the size of the application, you can use the SDK runtime master installer for online installation. The installer downloads the required modules from the Intel website during installation. Use the `--finstall=<feature-list> --fnone=all` command line to customize the algorithms to be installed. See [Features and Components](#)<sup>29</sup> for the feature list.

## Uninstallation

The SDK runtimes are shared among multiple applications. The application installer should never remove any SDK runtimes. Leave them on the system.


The user can manually uninstall the SDK runtime from Control Panel → Programs and Features. Do so only after all SDK applications are uninstalled from the system.

## Speech Runtime and Language Packs

The speech runtime and language packs do not have an offline installation option. They are available only as online download if not already on the system.

To add the speech runtime and the language pack(s), append the speech runtime and language pack feature names to the installer option `--finstall`. See [Features and Components](#)<sup>29</sup> for the feature names. For example, use `--finstall=voice,genie_en_us` to download and install the speech runtime and the US English language pack.

If you use the [Module Offline Installation](#)<sup>25</sup> installation option, select the speech components and follow the instruction in `readme_cmd.txt`. For example, if you select the hand tracking module, the speech runtime and the US English language pack, the instruction in `readme_cmd.txt` is similar to `--finstall=core,hand,voice,genie_en_us`. The generated customized installer contains the local hand tracking runtime and the code to trigger downloading of the speech runtime and the language pack.

 During the runtime installation, due to license restriction, the installer needs to validate the Intel® RealSense™ 3D Camera platform. You need to prompt the user to plug in the camera if the camera is not integrated. The installer returns status code 3020 (see [Command Option and Status Code](#)<sup>27</sup>) and skips the speech component installation if the installer fails to validate the platform.

## 7.1.1 Command Option and Status Code

### Installer Command Option

You can use the following command line options with the SDK runtime installers:

<code>--silent --no-progress --acceptlicense=yes</code>	Run the installer in the silent mode.
<code>--pre-bundle=&lt;dir&gt;</code>	Create an installer under <dir> that installs the selected SDK features. <dir> must be an absolute path.
<code>--finstall=&lt;feature-list&gt; --fnone=all</code>	Install only the requested features. Use comma to separate multiple features. See <a href="#">Features and Components</a> <sup>29</sup> for the feature definitions.
<code>--finstall=all</code>	Install all runtimes.
<code>--extract-to=&lt;dir&gt;</code>	Unpack to the specified directory. The default directory is %TEMP%\{GUID}.


### Installer Status Code

The SDK runtime installers return the following status codes to the application installer:

Status Code	Description
0	The installation was completed successfully.
1	The installation was aborted.
2	The installation was canceled by the user.
3	The installation was completed with some errors in the optional component installation.
1633	The installation was aborted due to unsupported platform or OS.
3010	The installation was completed and reboot was required. Reserved. Usually the SDK runtime does not require reboot.
3015	The installation was completed with the warning that the Intel® RealSense™

Depth Camera Manager installed on the system was incompatible with the SDK runtime version. The user must upgrade the Intel RealSense Depth Camera Manager for the SDK runtime to function properly.

- 3020 The installer failed to identify the platform as a valid Intel® RealSense™ 3D Camera platform. Premium components (that had license restrictions) were not installed. Other component installation was completed successfully.

 See the installer log under `%TEMP%/micl_tmp_<username>` for the detailed error description.


## 7.1.2 Installation Features

To describe different installation items, the SDK runtime installer uses the concept `Feature` to describe the set of installations that enable certain SDK functionality. For example, the `core` feature installs the runtime libraries needed for color and depth data capturing. Each feature include the 32-bit and 64-bit runtime dynamic libraries and the data files shared between 32-bit and 64-bit applications.

Use `Feature` in the following cases:

- **In the application installer:** For example, specify `--install=hand` to install the hand tracking module: its runtime libraries and data files.
- **With the `RealSenseInfo` JavaScript:** For example, use `RealSenseInfo(['voice', 'genie_en_us'], callback)` to check and install the speech recognition and synthesis runtime with the US English language pack.

Camera	Feature Name	Feature Description
Both	<code>core</code>	The SDK essentials including camera data capturing. This is must have for any SDK applications.
F200	<code>blob</code>	The blob/contour extraction algorithm.
F200	<code>emotion</code>	The emotion detection algorithm and data files.
R200	<code>enhanced_photography</code>	The enhanced photography algorithms.
Both	<code>face3d</code>	The face tracking algorithm and data files.
F200	<code>hand</code>	The hand tracking algorithm and data files.
F200	<code>personify</code>	The user segmentation algorithm.
R200	<code>scene_perception</code>	The scene perception algorithm and data files.
Both	<code>scan3d</code>	The 3D scanning algorithm and data files.
F200	<code>touchlesscontroller</code>	The touchless controller algorithm and data files.
F200	<code>track3d_metaio</code>	object tracking algorithm.
F200	<code>utils</code>	The SDK utility functions such as data smoothing.

Both	voice	<p>The speech recognition and synthesis algorithms.</p> <p> The speech recognition feature requires at least one speech language pack to be installed on the system.</p>
	genie_en_us	The US English language pack.
	genie_en_gb	The British English language pack.
	genie_es_la	The Latin American Spanish language pack.
	genie_pt_br	The Portuguese language pack.
	genie_de_de	The German language pack.
	genie_fr_fr	The French language pack.
	genie_it_it	The Italian language pack.
	genie_ja_jp	The Japanese language pack.
	genie_zh_cn	The Chinese language pack.

## 7.2 Supporting Libraries

The SDK provides the following supporting libraries that you should deploy them within the application package:

Supporting Libraries	Description
<b>\$ (RSSDK_DIR) /bin/win32:</b> libpxcclr.cs.dll libpxcpp2c.dll  <b>\$ (RSSDK_DIR) /bin/x64:</b> libpxcclr.cs.dll libpxcpp2c.dll	The supporting libraries for C#.
<b>\$ (RSSDK_DIR) /bin/win32:</b> libpxcclr.unity.dll libpxccpp2c.dll libpxccpp2c.dll.signature  <b>\$ (RSSDK_DIR) /bin/x64:</b> libpxcclr.unity.dll libpxccpp2c.dll libpxccpp2c.dll.signature	The supporting libraries for the Unity PC standalone platform.
<b>\$ (RSSDK_DIR) /framework/common/JavaScript:</b> realsenseinfo.js	The supporting libraries for the Unity web player platform.
<b>\$ (RSSDK_DIR) /framework/common/pxcclr.java/bin/x64:</b> libpxcclr.java.jar libpxcclr.jni.dll	The supporting libraries for Java.
<b>\$ (RSSDK_DIR) /framework/common/pxcclr.java/bin/x64:</b> libpxcclr.processing.jar libpxcclr.jni.dll	The supporting libraries for Processing.
<b>\$ (RSSDK_DIR) /framework/common/JavaScript:</b> realsense.js realsenseinfo.js	The supporting libraries for JavaScript.

## 7.3 Checking Module Availability

Before using a module, you should check module availability. Some modules may not be available on the target system due to licensing restrictions.

To check if a module is functional after installation, you can try creating an instance of the module in your application. In Example 1, the application creates an instance of the default module. If the instance is successfully created, the module is functional. Otherwise, the application may need to warn the end-user about any licensing restriction.

### C++ Example 1: Check Module Availability

```
bool CheckModuleStatus(pxcUID cuid) {
    PXCBBase *instance=0;
    PXCSession *session=PXCSession::CreateInstance();
    if (session) {
        if (session->CreateImpl(cuid,&instance)>=PXC_STATUS_NO_ERROR)
            instance->Release();
        session->Release();
    }
    return instance!=0;
}
```

### C# Example 1: Check Module Availability

```
Boolean CheckModuleStatus(Int32 cuid) {
    PXCMBase instance=null;
    PXCMSession session=PXCMSession.CreateInstance();
    if (session!=null) {
        if (session.CreateImpl(cuid,out instance)
>=pxcmStatus.PXCM_STATUS_NO_ERROR)
            instance.Dispose();
        session.Dispose();
    }
    return instance!=null;
}
```

### Java Example 1: Check Module Availability

```
boolean CheckModuleStatus(int cuid) {
    PXCMBase instance=null;
    PXCMSession session=PXCMSession.CreateInstance();
    if (session!=null) {
        instance=new PXCMBase();
        if (session.CreateImpl(cuid, instance)
>=pxcmStatus.PXCM_STATUS_NO_ERROR)
            instance.close();
        session.close();
    }
    return instance!=null;
}
```



## 7.4 Checking Camera Driver Version

The SDK capture module is where the accesses to the camera are implemented. Sometimes, it is necessary to check the camera driver version to ensure certain feature is available. The capture module version is the same as the camera driver version. Thus you should check the capture module version instead, as illustrated in Example 2:

### C++ Example 2: Check Camera Driver Version

```
PXCSession::ImplVersion GetDriverVersion(PXCSession *session, PXCapture
*capture) {
    PXCSession::ImplDesc mdesc={};
    session->QueryModuleDesc(capture, &mdesc);
    return mdesc.version;
}
```

### C# Example 2: Check Camera Driver Version

```
PXCMSession.ImplVersion GetDriverVersion(PXCMSession session, PXCMCapture
capture) {
    PXCMSession.ImplDesc mdesc;
    session.QueryModuleDesc(capture, out mdesc);
    return mdesc.version;
}
```

### Java Example 2: Check Camera Driver Version


```
PXCMSession.ImplVersion GetDriverVersion(PXCMSession session, PXCMCapture
capture) {
    PXCMSession.ImplDesc mdesc=new PXCMSession.ImplDesc();
    session.QueryModuleDesc(capture, mdesc);
    return mdesc.version;
}
```

## 7.5 Privacy Requirements and Guidelines

The SDK license exhibit A contains a set of requirements and guidelines for meeting end user privacy expectations in applications that utilize the SDK. This section of the document provides some additional information to assist in meeting these expectations.


There are several types of data produced by the SDK that could potentially contain personally identifiable information (PII). The five data types are:

- Raw color or depth data from the camera
- Raw microphone data
- Face recognition data
- Face-based emotion detection data
- Face-based age detection data

 In the future, there may be additional types of data that could contain PII so it is important to review this document in each new release of the SDK.

The SDK license requires that any application that directly accesses and processes any PII provide a notice to the end user when one or more of these data types are processed and must explain what actions are intended. The actions could include:

- Store the data on the local device
- Transmit the data over the internet
- Store the data on a remote server
- Share the data with a third party

 An explicit opt-in must be obtained from the end user if the data is to be shared with a third party.

The sections "[Example of Privacy Notification Page](#)"<sup>36</sup> and "[Example of Data Type Description](#)"<sup>35</sup> provide example implementations.

### 7.5.1 Example of Data Type Description

The privacy notification must use descriptive language to explain the PII data types and why they are of concern to the end user. Table 6 shows sample data type descriptions for the privacy notification.

Raw Camera Data	The application will have access to the raw picture data coming from the camera. This includes a color image and a depth image.
Raw Microphone Data	The application will have access to the raw sound data coming from the microphone.
Face Recognition	The application will identify you based on your face. Your identity will be stored as your name and user ID.
Emotion Detection	The application will have access to your emotional state, as determined by pictures of your face.
Age Detection	The application will have access to your age, as determined by pictures of your face.
Gender Detection	The application will have access to your gender, as determined by pictures of your face.
Face Detection	The application will have access to information about how many different faces are in the camera field of view.

**Table 6: Descriptive PII Data Type Language**

## 7.5.2 Example of Privacy Notification Page

An example notification page that could be displayed to end users is provided in Figure 4. This information should be displayed every time a new user is running the application for the first time. Another option is to display the information during installation and then include a link to the details every time a new user is running the application for the first time.



Figure 4: Example Privacy Notification Page

## 8 Working with Multiple Modalities

The SDK includes multiple algorithms (modalities or modules, used interchangeably) and each extends your application's interaction with the user. When enabling multiple modalities in your application, you must observe the following limitations/considerations:


### Stream Resolution and Frame Rate

Multiple modalities must agree on the stream resolutions and frame rates. For example, a face tracking module that needs color 1920x1080x30fps cannot work together with the object tracking module that works only with color 640x480x30fps. An agreed upon stream configuration must be set for the two modules to function together. The configuration is stream based thus a module that uses the color stream does not conflict with a module that works with a depth stream.

The following table shows the modules and their available configurations:

Camera F200		Color Resolution					Depth Resolution					
		width		320	640	640	960	1280	1920	Depth Resolution		
		height	240	360	480	540	720	1080	240	480	640	
Face Tracking (2D)				X				NA	NA	NA		
Face Tracking (3D)			X	X	X	X	X	X	X	X		
Hand Tracking		NA	NA	NA	NA	NA	NA	X	X	X		
User segmentation		X	X	X	X	X			X			
Emotion Detection		X		X				NA	NA	NA		
Object Tracking (2D)		X	X	X	X	X		NA	NA	NA		
Object Tracking (3D)		X	X	X	X	X			X			
Touchless Controller		NA	NA	NA	NA	NA	NA	X	X	X		
3D Scanning					X				X			
Camera R200		Color Resolution					Depth Resolution					
		width		320		640		1280	1920	320	480	628
		height	240	480		720		720	240	360	468	
Face Tracking			X			X				X		
Enhanced Photography		X		X		X	X					

Scene Perception	X		X				X	X	X
3D Scanning			X					X	

 This table is for reference only and may change over SDK releases. You should not hard code the settings. Instead, use `SenseManager` to auto-negotiate a working configuration.

The `PXC[M]SenseManager` interface auto-negotiates the stream configuration when you enable multiple modalities. The `Init` function returns successfully if there is an agreed upon stream configuration. Example 3 shows how to enable face and hand modules and auto-negotiate the configuration.

### C++ Example 3: Enable Two Modalities and Auto-Negotiate Configurations

```
// Create a SenseManager instance.
PXC_SenseManager *sm=PXC_SenseManager::CreateInstance();

// Enable face & hand tracking.
sm->EnableFace();
sm->EnableHand();

// additional face and hand configuration.
...

// Init
pxcStatus sts=sm->Init();
if (sts>=PXC_STATUS_NO_ERROR) {
    // two modalities can work together.
} else {
    // conflict in modalities.
}
```

### C# Example 3: Enable Two Modalities and Auto-Negotiate Configurations

```
// Create a SenseManager instance.
PXCMSenseManager sm=PXCMSenseManager.CreateInstance();

// Enable face & hand tracking.
sm.EnableFace();
sm.EnableHand();

// additional face and hand configuration.
...

// Init
pxcmStatus sts=sm.Init();
if (sts>=pxcmStatus.PXCM_STATUS_NO_ERROR) {
    // two modalities can work together.
} else {
    // conflict in modalities.
}
```

### Java Example 3: Enable Two Modalities and Auto-Negotiate Configurations

```
// Create a SenseManager instance.
PXCM_SenseManager sm=PXCM_SenseManager.CreateInstance();

// Enable face & hand tracking.
sm.EnableFace();
sm.EnableHand();

// additional face and hand configuration.
...

// Init
pxcmStatus sts=sm.Init();
if (sts>=pxcmStatus.PXCM_STATUS_NO_ERROR) {
    // two modalities can work together.
} else {
    // conflict in modalities.
}
```

## Device Properties

The camera device exposes many device properties to adjust the behavior of the capturing process. These device properties affect the captured image qualities. Different SDK modalities work best under different sets of device properties, due to the nature of the algorithms. For example, using a deeper smoothing setting smooths the captured images thus is good for modalities that need smoother edges and clean images. Since the local details are lost, such setting is not good for modalities that need to detect and respond to fast local movements.

For an application that works with a single modality, with a bit of trial and error, it is possible to find the setting that works better than others. The following table shows the depth settings for different modalities:

Camera F200	Confidence Threshold	Filter Option	Laser Power	Motion Range Trade off	Accuracy
Factory Default	6	5	16	0	Median
Face Tracking	1	6	16	21	Median
Hand Tracking	1	6	16	0	Median
User Segmentation	0	6	16	21	Coarse
Object Tracking	6	5	16	0	Median
Touchless Controller	1	6	16	0	Median
3D Scanning	5	4	16	0	Finest



The table is shown only for reference. The settings may change (or relaxed) over different SDK releases as the modality algorithms improve.

When enabling multiple modalities, you need to carefully trade off what is most important, as it is often the case that the best setting of the modalities differs. You can set the device properties in the following ways:

- **Happy Median:** Choose the setting that may not be the best for each modality but good enough for the application context. This works when there is no dominant interaction. Each interaction must be accurate (unambiguous).
- **Dominant Instance:** Choose the setting that works best for the dominant interaction. For example, if the application context is to use face tracking to identify a region of interest on the screen and then use hand tracking to manipulate the objects in the region, it is possible to set the setting best for face tracking initially and switch to the setting best for hand tracking when needed.

See also [Sharing Device Properties](#)<sup>51</sup> for more details how to manage device properties.

## Power and Performance

Consider power and performance as well. Each SDK modality incurs nontrivial workloads, although the SDK continues to optimize for the platform. No data to share as this is highly platform and application context dependent. You must experiment with the actual workloads to understand your application needs on the target platform.

It is recommended to design your application context to time share multiple modalities, that is, to enable different modalities at different time. The user gets to enjoy different interactions thus keeps the user engaged. Furthermore, the user is less likely to get muscle fatigue by doing a single interaction too long. To do so, enable multiple modalities at initialization and then pause/resume the modalities, as illustrated in Example 4. The alternative is to close the `SenseManager` pipeline and reinitialize for each modality interaction, which works best if the application context switch is long enough to allow camera close and reopen.

### C++ Example 4: Pause Face Computation

```
// pp is a PXC_SenseManager instance.
pp->PauseFace(true);
...
pp->PauseFace(false);
```

### C# Example 4: Pause Face Computation

```
// pp is a PXCMSenseManager instance.
pp.PauseFace(true);
...
pp.PauseFace(false);
```

### Java Example 4: Pause Face Computation



```
// pp is a PXCMSenseManager instance.  
pp.PauseFace(true);  
...  
pp.PauseFace(false);
```

## Coordinate Systems

Finally, it is important to understand the coordinate system that each modality operates when there is a need to combine different modality data together. See [Coordinate Systems](#) for the coordinate system definitions. The following table shows the coordinate systems that the modalities operate on for reference:

	Coordinate System Used
Face Tracking (2D)	Use the color image coordinates.
Face Tracking (3D)	Use the camera coordinate system (depth sensor origin).
Hand Tracking	Use the camera coordinate system (depth sensor origin).
Object Tracking	Use the camera coordinate system (color sensor origin).

For example, you may use hand tracking to identify the hand position and use object tracking to locate a virtual object. You must use the `PXC[M]Projection` interface to map the coordinates of the output data to the same coordinate system, or you would find that the hand and the virtual object are at two distinct places. Example 5 shows how to do color/depth coordinates conversion.

### C++ Example 5: Color/Depth Coordinates Conversion

```
// device is a PXCCapture::Device instance
PXCCProjection *projection=device->CreateProjection();

// Convert from depth(u,v,z) to color(i,j)
{
    PXCPPoint3DF32 depth = {u, v, z};
    PXCPPointF32 color;
    projection->MapDepthToColor(1, &depth, &color);
    i = color.x;
    j = color.y;
}

// Convert from color(i,j) to depth(u,v)
// sample is a PXCCapture::Sample instance
{
    PXCPPointF32 color = {x, y};
    PXCPPointF32 depth;
    projection->MapColorToDepth(sample->depth, 1, &color, &depth);
    u = depth.x;
    v = depth.y;
}

// Clean up
projection->Release();
```

### C# Example 5: Color/Depth Coordinates Conversion

```
// device is a PXCMCapture.Device instance
PXCMProjection projection=device.CreateProjection();

// Convert from depth(u,v,z) to color(i,j)
{
    PXCMPoint3DF32[] depth=new PXCMPoint3DF32[1];
    depth[0].x=u;
    depth[0].y=v;
    depth[0].z=z;
    PXCMPointF32[] color;
    projection.MapDepthToColor(depth, out color);
    i = color[0].x;
    j = color[0].y;
}

// Convert from color(i,j) to depth(u,v)
// sample is a PXCMCapture.Sample instance
{
    PXCMPointF32[] color = new PXCMPointF32[1];
    color[0].x=x;
    color[0].y=y;
    PXCMPointF32[] depth;
    projection.MapColorToDepth(sample.depth, color, out depth);
    u = depth[0].x;
    v = depth[0].y;
}

// Clean up
projection.Dispose();
```

### Java Example 5: Color/Depth Coordinates Conversion

```
// device is a PXCMCapture.Device instance
PXCMProjection projection=device.CreateProjection();

// Convert from depth(u,v,z) to color(i,j)
{
    PXCMPoint3DF32[] depth=new PXCMPoint3DF32[1];
    depth[0]=new PXCMPoint3DF32();
    depth[0].x=u;
    depth[0].y=v;
    depth[0].z=z;
    PXCMPointF32[] color=new PXCMPointF32[1];
    PXCMPointF32[0]=new PXCMPoint3DF32();
    projection.MapDepthToColor(depth, color);
    i = color[0].x;
    j = color[0].y;
}

// Convert from color(i,j) to depth(u,v)
// sample is a PXCMCapture.Sample instance
{
    PXCMPointF32[] color = new PXCMPointF32[1];
    color[0]=new PXCMPointF32();
    color[0].x=x;
    color[0].y=y;
    PXCMPointF32[] depth=new PXCMPointF32[1];
    PXCMPointF32[0]=new PXCMPointF32();
    projection.MapColorToDepth(sample.depth, color, depth);
    u = depth[0].x;
    v = depth[0].y;
}

// Clean up
projection.close();
```

## 9 Coexisting with Other Applications

In a typical user system, multiple SDK applications may run concurrently competing for the physical I/O devices. For example, an SDK-enabled game application may compete the camera resource with an SDK-enabled background service. In this case, it is often impractical to ask the user to physically close one application before launching the other, especially when the application is a background service that is "invisible" to the user.


The following issues must be resolved:

- During setup, how do multiple applications negotiate a common device configuration? See [Sharing Device Configuration](#)<sup>45</sup> for details.
- During streaming, the user may switch among applications. How do applications maintain the optimal device settings for the application context? See [Sharing Device Properties](#)<sup>51</sup> for details.
- How do multiple applications share the platform computation resource? See [Sharing Platform Computations](#)<sup>53</sup> for details.

## 9.1 Sharing Device Configuration

The SDK provides limited support of multiple application access usage scenarios:

- Two or more applications access to the same device, using the same or compatible configuration. For example, two applications can co-access the device data if application 1 streams color at 640x480x30fps and application 2 streams color at 640x480x30fps and depth at 640x480x30fps.
- Specially developed SDK applications that can adapt to whatever configuration other applications use.

 Here device configuration refers to stream resolutions, frame rates and pixel format, excluding any device properties such as color brightness and hues. Device properties are unmanaged resource. See [Sharing Device Properties](#)<sup>51</sup> for recommendations how to manage the device properties.

The SDK introduces the concept of static application and adaptive application to describe different application scenarios and how they interact with each other.


### Static Application

A static application does not explicitly handle device configuration change during streaming. (The static application may work with multiple resolutions during initialization.) This is the default setting.

Static applications can configure the I/O device if there are no other static applications in active streaming. The first static application can retrieve a full enumeration of available configurations. The second or any other applications sees only the active configuration used by the first application.

Static applications have precedents over [adaptive applications](#)<sup>47</sup>. If a static application is launched after an adaptive application, the static application can set the device configuration, and the adaptive application receives a device configuration change event.

During streaming, static applications can expect steady streams of data without any interruption from other static or dynamic application activities.

 Non-SDK applications that access the same device may interrupt streaming. Since static applications do not handle configuration change, such interruption breaks the streaming loop. It is recommended that the application implement an outer loop to reinitialize and re-start streaming, as illustrated in Example 6.

C++ Example 6: Restart Streaming Upon Interruption

```
// pp is a PXC_SenseManager instance

// An outer loop to handle resolution change
pxcStatus sts=PXC_STATUS_STREAM_CONFIG_CHANGED;
while (sts==PXC_STATUS_STREAM_CONFIG_CHANGED) {
    // SenseManager initialization
    pp->EnableXXX();
    sts=pp->Init();
    if (sts<PXC_STATUS_NO_ERROR) break;

    // SenseManager streaming
    for (;;) {
        sts=pp->AcquireFrame();
        if (sts<PXC_STATUS_NO_ERROR) break;
        ...
        pp->ReleaseFrame();
    }
    pp->Close();
}
```

### C# Example 6: Restart Streaming Upon Interruption

```
// pp is a PXCM_SenseManager instance

// An outer loop to handle resolution change
pxcmStatus sts=pxcmStatus.PXCM_STATUS_STREAM_CONFIG_CHANGED;
while (sts==pxcmStatus.PXCM_STATUS_STREAM_CONFIG_CHANGED) {
    // SenseManager initialization
    pp.EnableXXX();
    sts=pp.Init();
    if (sts<pxcmStatus.PXCM_STATUS_NO_ERROR) break;

    // SenseManager streaming
    for (;;) {
        sts=pp.AcquireFrame();
        if (sts<pxcmStatus.PXCM_STATUS_NO_ERROR) break;
        ...
        pp.ReleaseFrame();
    }
    pp.Close();
}
```

### Java Example 6: Restart Streaming Upon Interruption

```
// pp is a PXCMSenseManager instance

// An outer loop to handle resolution change
pxcmStatus sts=pxcmStatus.PXCM_STATUS_STREAM_CONFIG_CHANGED;
while (sts==pxcmStatus.PXCM_STATUS_STREAM_CONFIG_CHANGED) {
    // SenseManager initialization
    pp.EnableXXX();
    sts=pp.Init();
    if (sts.comparedTo(pxcmStatus.PXCM_STATUS_NO_ERROR)<0) break;

    // SenseManager streaming
    for (;;) {
        sts=pp.AcquireFrame();
        if (sts.comparedTo(pxcmStatus.PXCM_STATUS_NO_ERROR)<0) break;
        ...
        pp.ReleaseFrame();
    }
    pp.Close();
}
```

## Adaptive Application

An adaptive application responds to the device configuration change event and adjusts its behaviors accordingly. The changed behavior could be:

- The application may restart with the new configuration.
- The application may pause its operation until a new suitable configuration is available.

It is recommended that any service applications handle device configuration change.

Adaptive applications can set the device configuration if there is no other application competing for the device. Otherwise, the adaptive application must use the active configuration. The application receives the `PXC[M]_STATUS_CONFIG_CHANGED` event when there is a configuration change during streaming.

You can use the `SetDeviceAllowProfileChange` function to instruct the device to operate in the adaptive mode. The device will return the `PXC[M]_STATUS_CONFIG_CHANGED` status code when there is a configuration change. Example 7 illustrates the concept. For simplicity, the example simply restarts the streaming loop (as in Example 6) when re-configuration occurs.

### C++ Example 7: Adaptive Application

```
// pp is a PXC_SenseManager instance

class MyHandler: public PXC_SenseManager::Handler {
public:
    virtual pxcStatus PXC_API OnConnect(PXCCapture::Device *device, pxcBool
connected) {
        if (connected) device->SetDeviceAllowProfileChange(true);
        return PXC_STATUS_NO_ERROR;
    }
};

// An outer loop to handle resolution change
pxcStatus sts=PXC_STATUS_STREAM_CONFIG_CHANGED;
while (sts==PXC_STATUS_STREAM_CONFIG_CHANGED) {
    // SenseManager initialization
    pp->EnableXXX();

    // Signal the device to operate in the adaptive mode.
    MyHandler handler;
    sts=pp->Init(&handler);
    if (sts<PXC_STATUS_NO_ERROR) break;

    // SenseManager streaming
    for (;;) {
        sts=pp->AcquireFrame();
        if (sts<PXC_STATUS_NO_ERROR) break;
        ...
        pp->ReleaseFrame();
    }
    pp->Close();
}
```

## C# Example 7: Adaptive Application



```
// pp is a PXCMSenseManager instance

pxcmStatus OnConnect(PXCMCapture.Device device, Boolean connected) {
    if (connected) device.SetDeviceAllowProfileChange(true);
    return pxcmStatus.PXCM_STATUS_NO_ERROR;
}

// An outer loop to handle resolution change
pxcmStatus sts=pxcmStatus.PXCM_STATUS_STREAM_CONFIG_CHANGED;
while (sts==pxcmStatus.PXCM_STATUS_STREAM_CONFIG_CHANGED) {
    // SenseManager initialization
    pp.EnableXXX();

    // Instruct the device to operate in the adaptive mode
    PXCMSenseManager.Handler handler=new PXCMSenseManager.Handler();
    handler.onConnect=OnConnect;
    sts=pp.Init(handler);
    if (sts<pxcmStatus.PXCM_STATUS_NO_ERROR) break;

    // SenseManager streaming
    for (;;) {
        sts=pp.AcquireFrame();
        if (sts<pxcmStatus.PXCM_STATUS_NO_ERROR) break;
        ...
        pp.ReleaseFrame();
    }
    pp.Close();
}
```

### Java Example 7: Adaptive Application

```
// pp is a PXCMSenseManager instance

class MyHandler implements PXCMSenseManager.Handler {
public:
    @override pxcmStatus OnConnect(PXCMCapture.Device device, boolean
connected) {
        if (connected) device.SetDeviceAllowProfileChange(true);
        return pxcmStatus.PXCM_STATUS_NO_ERROR;
    }
};

// An outer loop to handle resolution change
pxcmStatus sts=pxcmStatus.PXCM_STATUS_STREAM_CONFIG_CHANGED;
while (sts==pxcmStatus.PXCM_STATUS_STREAM_CONFIG_CHANGED) {
    // SenseManager initialization
    pp.EnableXXX();

    // Instruct the device to operate in the adaptive mode
    MyHandler handler=new MyHandler();
    sts=pp.Init(handler);
    if (sts.comparedTo(pxcmStatus.PXCM_STATUS_NO_ERROR)<0) break;

    // SenseManager streaming
    for (;;) {
        sts=pp.AcquireFrame();
        if (sts.comparedTo(pxcmStatus.PXCM_STATUS_NO_ERROR)<0) break;
        ...
        pp.ReleaseFrame();
    }
    pp.Close();
}
```

## 9.2 Sharing Device Properties

The device properties (such as color stream brightness, hue, and sharpness) are unmanaged resource in SDK 4.0.0. The applications must behave responsibly to achieve an optimal platform level user experience.

Use the following recommendations when designing your application:

- a) **SDK application in window focus:** Whenever the user switches your application to window focus, you must set (reset) the device properties to what your application must use. Certain device property has side effect. For example, turning on color stream auto exposure may lower the color stream frame rate in the low-light condition. To minimize such side effect, always reset all available device properties, as illustrated in Example 8.

### C++ Example 8: Restore Device Properties Upon Receiving Windows Focus


```
// device is a PXCCapture::Device instance
device->ResetPropertiesUponFocus();
```

### C# Example 8: Restore Device Properties Upon Receiving Windows Focus

```
// device is a PXCMCapture.Device instance
device.ResetPropertiesUponFocus();
```

### Java Example 8: Restore Device Properties Upon Receiving Windows Focus

```
// device is a PXCMCapture.Device instance
device.ResetPropertiesUponFocus();
```

 Upon initialization, you cannot assume that the device properties are at their default values. Set them to default values explicitly if this is critical for your application. Example 9 shows how to set depth stream options to default values.

### C++ Example 9: Reset Depth Stream Options to Default

```
// device is a PXCCapture::Device instance
device->ResetProperties(PXCCapture::STREAM_TYPE_DEPTH);
```

### C# Example 9: Reset Depth Stream Options to Default

```
// device is a PXCMCapture.Device instance
device.ResetProperties(PXCMCapture.StreamType.STREAM_TYPE_DEPTH);
```

### Java Example 9: Reset Depth Stream Options to Default

```
// device is a PXCMCapture.Device instance
device.ResetProperties(PXCMCapture.StreamType.STREAM_TYPE_DEPTH);
```

- b) **SDK application out of window focus:** Do not change any device properties while your application is not in window focus. Doing so may impact the quality/performance of the

---

SDK application in focus. You may read the device properties to determine if they are still suitable for your algorithms. If the answer is negative, turn off your algorithms.

- c) **SDK background applications or services:** Any SDK background application should not change the device properties unless there are other mechanisms to ensure that there is no active SDK application at the moment. In certain cases, for example, an authentication application may need to access to the camera exclusively for a period of time. Do so by starting a foreground window that takes the window focus. Then set the device properties. This is similar to the behavior of any regular SDK application that is in window focus.

## 9.3 Sharing Platform Computations

Although the SDK is optimized, there are still non-trivial workloads in module algorithm processing. The applications must behave responsibly so that the overall platform experience is not compromised. The recommendation is as follows concerning three types of SDK applications:

- a) **SDK application in window focus:** Any SDK application with its window in focus can assume priority in computing resources. The application can use any SDK resources for the best user experience.
- b) **SDK application out of window focus:** If the SDK application loses its window focus, the application should behave responsibly and yield computation resources to any foreground application in focus. The application should pause any non-trivial SDK module processing, as illustrated in Example 10.

### C++ Example 10: Pause Face Computation

```
// pp is a PXCSceneManager instance.  
bool pause=IsWindowFocusLost();  
pp->PauseFace(pause);
```

### C# Example 10: Pause Face Computation

```
// pp is a PXCSceneManager instance.  
Boolean pause=IsWindowFocusLost();  
pp.PauseFace(pause);
```

### Java Example 10: Pause Face Computation

```
// pp is a PXCSceneManager instance.  
boolean pause=IsWindowFocusLost();  
pp.PauseFace(pause);
```


- c) **SDK background applications or services:** Any SDK background applications or services must be specifically designed such that they are in the idling state most of the time and only become active when certain designated events trigger. When they are in the idling state, they do not consume significant computation. When they become active, they perform their functions and then go back to the idle state.

## 10 Managing Power and Performance

Managing power and performance is critical to deliver the best user experience. The SDK uses a simple concept to help manage your application power and performance:

- You can explicitly set the power state: performance or battery. The performance mode instructs the algorithms and the camera to run as best experience as the platform can provide. The battery mode yields the control to the algorithms or the camera to determine if there is chance to save power, by lowering the frame rate, or using a different algorithms etc.
- In the battery mode, the algorithm(s) may suggest the power state. For example, the hand module may suggest (after some computation) that there is no hand in the scene. Thus it is safe to go into the battery mode. The face module may suggest to go into the battery mode if there is no face in the scene. If any of the sources suggests the performance mode, the SDK is in the performance mode. Otherwise, the SDK is in the battery mode.

The power state is managed by the `PXC[M]PowerState` interface. Example 11 shows an example of how to set the power state to be in the performance mode.

 The default state is the battery mode. You want to keep the system in the battery mode as much as possible. Changing the state to the performance mode only briefly when you need the best interaction experience.

### C++ Example 11: Manage Power State and Inactivity Interval

```
// ss is a PXCSession instance.
PXCPowerState *ps=ss->CreatePowerManager();

// Set the power state
ps->SetState(PXCPowerState::STATE_PERFORMANCE);

// Set the inactivity interval.
ps->SetInactivityInterval(5);

// Clean up
pp->Release();
```

### C# Example 11: Manage Power State and Inactivity Interval

```
// ss is a PXCMSession instance.
PXCMPowerState ps=ss.CreatePowerManager();

// Set the power state
ps.SetState(PXCMPowerState.State.STATE_PERFORMANCE);

// Set the inactivity interval.
ps.SetInactivityInterval(5);

// Clean up
pp.Dispose();
```

### Java Example 11: Manage Power State and Inactivity Interval

```
// ss is a PXCMSession instance.
PXCMPowerState ps=ss.CreatePowerManager();

// Set the power state
ps.SetState(PXCMPowerState.State.STATE_PERFORMANCE);

// Set the inactivity interval.
ps.SetInactivityInterval(5);

// Clean up
pp.close();
```

## 11 Algorithm Operating Ranges

The following table summarizes the operating range of the SDK algorithms:

Algorithm	Camera F200		Camera R200	
	Detection (cm)	Tracking (cm)	Detection (cm)	Tracking (cm)
Face: Detection	25-75	30-100	50-250	55-250
Face: Landmark	30-100	30-100	50-150	50-150
Face: Recognition	30-80	30-80	30-80	30-80
Face: Expression	30-100	30-100	30-100	30-100
Face: Pulse	30-60	30-60	30-70	30-70
Face: Pose	30-100	30-100	50-150	50-150
Hand: Blob**	20-85	20-85	NA	
Hand: Segmentation**	20-60	20-60	NA	
Hand: Tracking**	20-60	20-60	NA	
Hand: Gesture**	20-60	20-60	NA	
Object Tracking	30-180*		NA	
Emotion	25-100		NA	

\* Require at least 5000 pixels at 640x480 for proper detection, which translates to

Distance	Minimum Rectangular Object Size
30 cm	4.5 cm x 3.5 cm
60 cm	9 cm x 7 cm
100 cm	16 cm x 11 cm
150 cm	23 cm x 18 cm
180 cm	28 cm x 21 cm

\*\* Require a minimum palm size of 5.5 cm (about a fix-year old kid palm size) and hand interaction speed within the following range:

Resolution	Speed Limitation
HVGA	2m/s
VGA	0.75m/s



