



# AppAble Library For C++

## Reference Guide

### Introduction:

The AppAble dll library allows C++ developers to make applications on their IDE and compiler of choice on the Windows platform that connect to Intel App store.

### Wrapper Contents:

- The Software License Agreement (License.txt)
- The AppAble Library files (Library)
- A guide to create the MSI file for submission
- A sample C++ project that connects to the store (Sample). Including:
  1. A C++ header file (baKnoDllLoader.h) that contains the functions needed to connect to Intel's Client Agent.
  2. A C++ code file (baKnoDllLoader.cpp) that implements the functions of the header file.

**Note:** The header and code files are free to the developer to modify to satisfy their needs. However, it is recommended no to in order to provide the critical functionality required to submit an application to Intel's App Store.

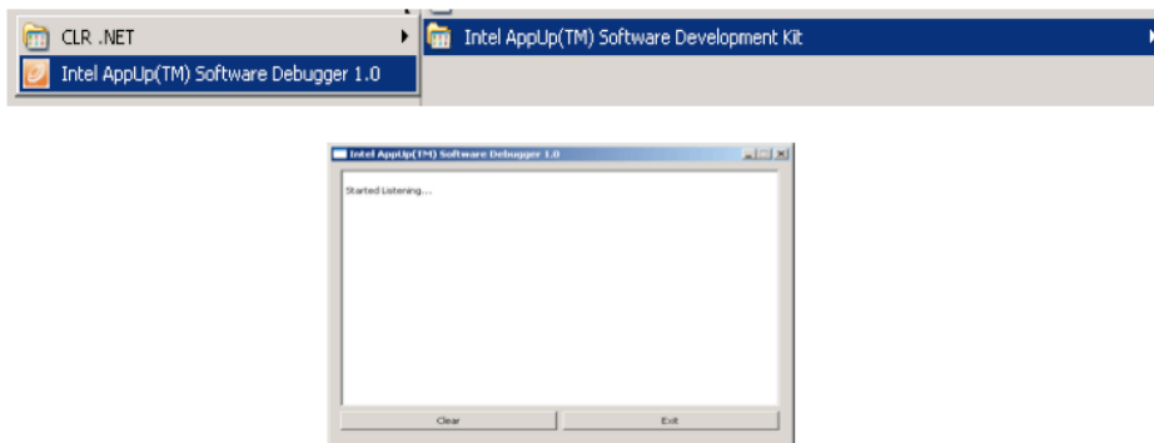
### Developer requirements:

- The Intel's App store SDK. <http://appdeveloper.intel.com/en-us/>
- Microsoft's runtime libraries (provided with the package).
- A development environment that uses C++ on a Windows platform.

### How to use the SDK:

There are two environments, Debug and Production. The idea is to develop under Debug and only when everything works, change to Production to build the final application and submit it to the AppUp store.

To test your Debug application you need to run the "Intel AppUp Software Debugger", an application that emulates a communication to the AppUp store. Then run the Debug application and check if all communication stages work as described on the docs. This Debugger is run by calling it from the Start Menu and going to the Intel SDK program folder.

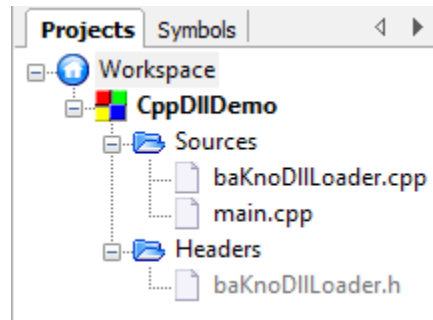


### Developing a Simple Console Application:

The example application does the following steps:

- Connects to the Client Agent
- Checks the Authorization status
- Begins an event
- Ends the event
- Disconnects from the Client Agent

First we need to create a simple console application project in any IDE. The sample application provided uses Code::Blocks as IDE and MingW as the c++ compiler.



Add the files “baKnoDllLoader.h” and “baKnoDllLoader.cpp” to the projects in the Headers and Sources folders respectively. As shown in the picture above.

Then we need to modify “main.cpp” in order to use the functions that connects the application to the Client agent. Start by including the library:

```
#include "baKnoDllLoader.h"
```

Then on the main function we start the connection to the Client Agent

```
int connection = ConnectToIntelAPI(0x11111111, 0x11111111, 0x11111111, 0x11111111,  
    "11111111-1111-1111-1111-111111111111");
```

If an error occurs, the error message can be obtained with the “GetErrorMessage” function, the integer result can be translated to a string for a better read for the end user.

To check the authorization status of our application we use the following line:

```
int message = CheckAuthorizationStatus();
```

Sometimes it is required to record use time. For that reason, two commands are present to do the job easily. Remember that “every beginning has an end”, it is recommended to use these two functions in pair.

```
int message = BeginRecordingEvent();
```

And

```
int message = EndRecordingEvent();
```

And to disconnect, the following function is executed

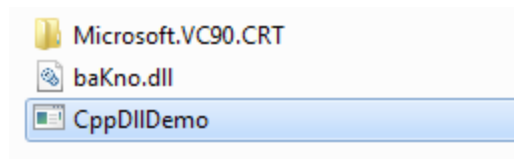
```
DisconnectFromIntelAPI();
```

The Intel AppUp SDK also offers an upgrade feature, to get other applications as well, this is present in our wrapper as well with the following function:

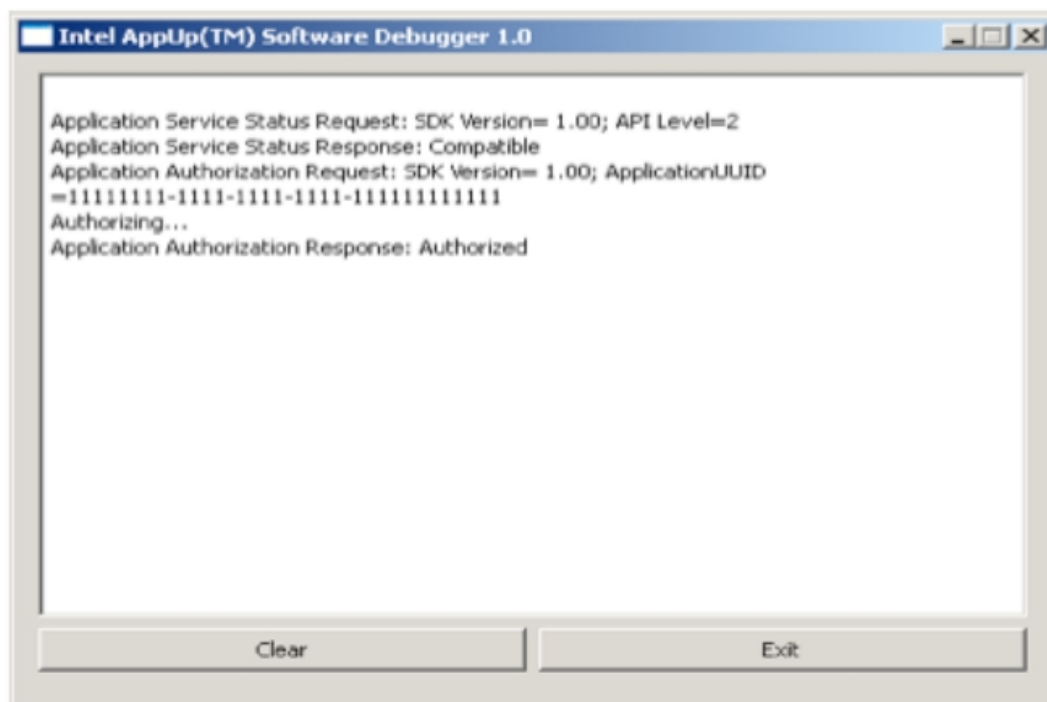
```
int message = Upgrade(0x22222222, 0x33333333, 0x44444444, 0x55555555);
```

## Deployment of the Application

When the application is compiled. Place the dll wrapper with the runtime libraries on the same folder where the Application's executable file is located:



Compile the file as “CppDIIDemo.exe”. Then the Debugger from the SDK must start before the demo is executed.



## List of Commands:

### 1. Connect to the App store:

```
int ConnectToIntelAPI(unsigned long l1, unsigned long l2,  
                     unsigned long l3, unsigned long l4, char *baKnoID)
```

This command initializes the connection to the App store. Each of the four first parameters are part of the GUID that the Intel developer program provides to the developer. The last parameter is the GUID provided by baKno for validations.

**Note:** Because of the current development status of the SDK, The parameter values must be in hexadecimal of size 8 exactly (i.e.:0x12345678).

So, if the id provided is: 12341234-2345-3456-4567-567890123456

The parameters are: 0x12341234, 0x23453456, 0x45675678 and 0x90123456

**Note:** For developer purposes, the SDK provides a debugging ID so developers can test their applications. Each parameter has the value 0x11111111. baKno has a debug ID as well. Thus, the command to connect to the developer's App store is:

```
int a = ConnectToIntelAPI(0x11111111, 0x11111111, 0x11111111, 0x11111111,  
                          "11111111-1111-1111-1111-111111111111");
```

#### Return values:

- **BKNO\_FAILURE:** The Application presented an Unknown Error.
- **BKNO\_SUCCESS:** The Application started successfully.
- **BKNO\_NOT\_INITIALIZED:** The application could not start.
- **BKNO\_NOT\_AVAILABLE:** The Intel's Client Agent is not Available.
- **BKNO\_NOT\_AUTHORIZED:** The Application is not authorized.
- **BKNO\_AUTHORIZATION\_EXPIRED:** The Application ID has expired.
- **BKNO\_TIME\_OUT:** The connection with the Client Agent reached a timeout.
- **BKNO\_INVALID\_ID:** The GUID provided for baKno's validation is invalid

## 2. Check the Application Status:

*int CheckAuthorizationStatus()*

Command that checks the current status of the Application

**Return Values:**

- **BKNO\_AUTHORIZED:** The Application is authorized
- **BKNO\_NOT\_AUTHORIZED:** The Application is unauthorized.
- **BKNO\_AUTHORIZATION\_EXPIRED:** The Application ID has expired.
- **BKNO\_TIME\_OUT:** The connection with the Client Agent reached a timeout.

## 3. Save The Application Runtime (Optional)

*int BeginRecordingEvent()*

Command that allows the App store client to start recording the Application's runtime.

*int EndRecordingEvent()*

Command that ends the recording of the Application's runtime.

These two commands allow the developer to obtain statistics about the time each customer runs the Application.

**Note:** Both commands MUST be used together, meaning, that with each time *BeginRecordingEvent* is used, there must be *EndRecordingEvent* command later on.

**Return Values:**

- **BKNO\_FAILURE:** The Application presented an unknown error.
- **BKNO\_SUCCESS:** The command has been executed successfully.
- **BKNO\_NOT\_AUTHORIZED:** The Application is not authorized.
- **BKNO\_AUTHORIZATION\_EXPIRED:** The Application ID has expired

- **BKNO\_NO\_APP\_BEGIN\_EVENT:** The “begin event” has not been called.
- **BKNO\_TIME\_OUT:** The connection with the Client Agent reached a timeout.

#### 4. End the Connection:

*void DisconnectFromIntelAPI()*

Command that terminates the connection to Intel’s App store.

**Note:** This command MUST be executed when the end user closes the Application. If it’s executed earlier, the application may close by itself immediately. This is due to the App store has a record of the application that is running and terminates the process automatically.

#### 5. Upgrade the Application

*int Upgrade(unsigned long l1, unsigned long l2, unsigned long l3, unsigned long l4)*

Command that allows the upgrade process from one application to another.

##### Return Values:

- **BKNO\_FAILURE:** The Application presented an unknown error.
- **BKNO\_SUCCESS:** The command has been executed successfully.
- **BKNO\_NOT\_AUTHORIZED:** The Application is not authorized.
- **BKNO\_AUTHORIZATION\_EXPIRED:** The Application ID has expired
- **BKNO\_TIME\_OUT:** The connection with the Client Agent reached a timeout.