



AppAble DLL Wrapper For C#

Reference Guide

Introduction:

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

Package Contents:

- The Software License Agreement (License.txt)
- The AppAble library files (Library)
- The dependency's libraries (Microsoft.VC90.CRT)
- A Folder containing a C# project that access the library (Sample)

Note: The C# file “baKnoDllReader.cs” is free to the developers to modify to satisfy their needs. However, it is recommended not to do so in order to provide the critical functionality required to submit an application to Intel's App Store. Also, you can convert the file to VisualBasic code and use it in your .Net application.

Developer requirements:

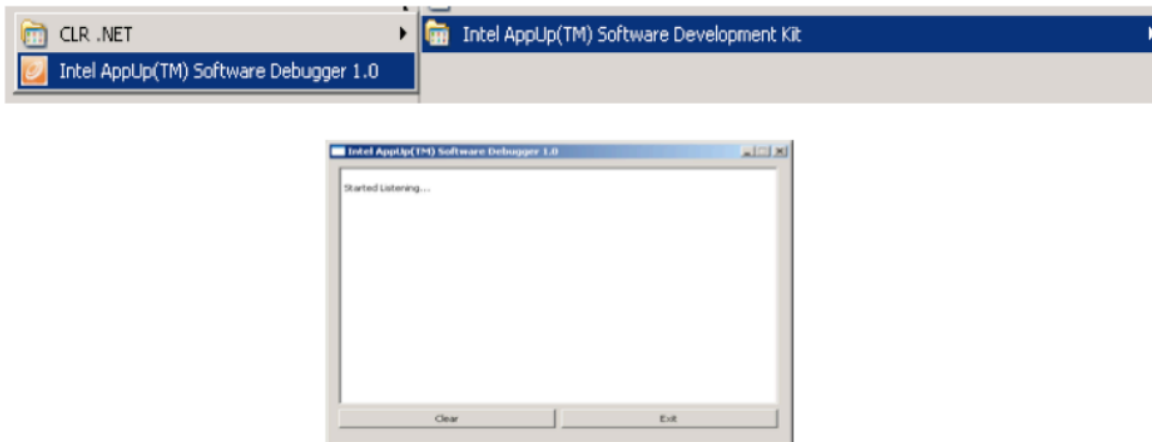
- The Intel’s App store SDK. It can be obtained here: <http://appdeveloper.intel.com/en-us/>

- Microsoft's runtime libraries (provided with the package).
- A development environment that uses C# (.Net or Mono).

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.

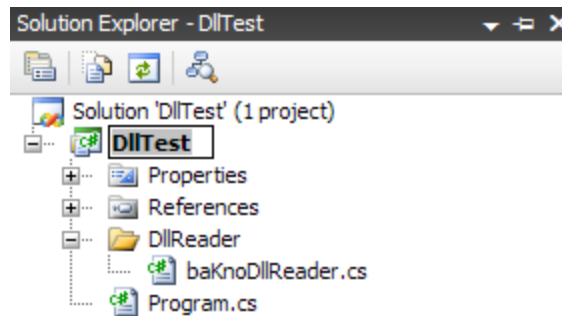


Developing a Simple Console Application:

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 Visual Studio 2008, but it can be done on any other that uses C#.



The next step is to add the file “baKnoDllReader.cs” to the project on the folder of preference as shown on the picture above.

Then we proceed to the main code file, in this case “Program.cs” and we use the wrapper code to access the functions.

```
using baKnoDLLWrapper;
```

Then we proceed to perform the connection to the Client Agent.

```
static void Main(string[] args)
{
    int connection = baKnoDllReader.ConnectToIntelADP(0x11111111, 0x11111111,
    0x11111111, 0x11111111, "11111111-1111-1111-1111-111111111111");
```

We need to check whether or not the application is authorized

```
if (connection != (int)baKnoDllReader.connResults.BKNO_SUCCESS && connection !=
(int)baKnoDllReader.connResults.BKNO_AUTHORIZED)
{
    Console.WriteLine(baKnoDllReader.GetErrorMessage(connection));
    Console.WriteLine("Closing Test Application...");
    System.Threading.Thread.Sleep(5000);
}
```

The static function “baKnoDllReader.GetErrorMessage” allows the developer to display an error message that is related to the code returned by the wrapper. It is not mandatory to use it.

To check the authorization status of the application we use the following command.

```
int status = baKnoDllReader.GetIntelADPStatus();
```

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.

```
status = baKnoDllReader.BeginIntelEvent();
```

And

```
status = baKnoDllReader.EndIntelEvent();
```

And to disconnect, the following function is executed

```
baKnoDllReader.DisconnectFromIntel();
```

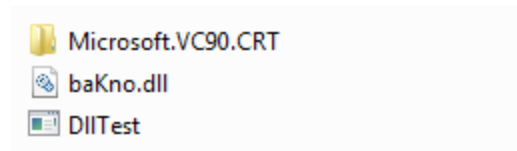
The Intel AppUp SDK also offers an upgrade feature, this is present in our wrapper as well with the following function:

```
status = baKnoDllReader.Upgrade(0x22222222, 0x33333333, 0x44444444, 0x55555555);
```

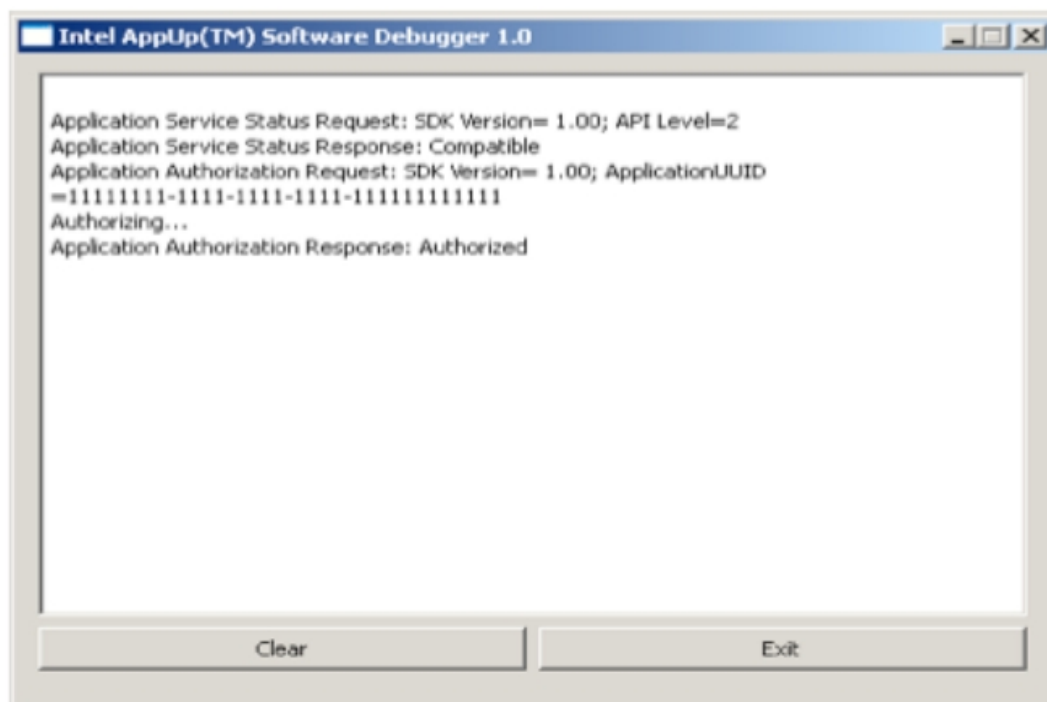
The full source code of the example is provided within this package.

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 “DLLTest.exe”. Then the Debugger from the SDK must start before the demo is executed.



List of Commands:

1. Connect to the App store:

int ConnectToIntelADP(uint l1, uint l2, uint l3, uint l4, string 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 baKno provides for validation.

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 = ConnectToIntelADP(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_ID_FAILURE:** The GUID provided by baKno is invalid.

2. Check the Application Status:

int GetIntelADPStatus()

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 BeginIntelEvent()

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

int EndIntelEvent()

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 *BeginIntelEvent* is used, there must be *EndIntelEvent* 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 DisconnectFromIntel()

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

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

5. Upgrade

int Upgrade(uint l1, uint l2, uint l3, uint l4)

Command that allows the application to upgrade to another application. The four parameters are the GUID of the new application to upgrade to.

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.