



# AppAble Library For Python

## Reference Guide

### Introduction:

The AppAble library allows Python developers to make applications on their IDE of choice on the Windows and Linux platforms that connect 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 Python application that access the library (Sample)
- A guide to create the MSI file for submission

**Note:** The Python compiled file "ATDSWrapper.pyc" must be included in the application. It contains a class with all critical functionalities required to submit an application to Intel's App Store. Also, keep the Libs folder next to this file, it contains the required libraries.

### Developer requirements:

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

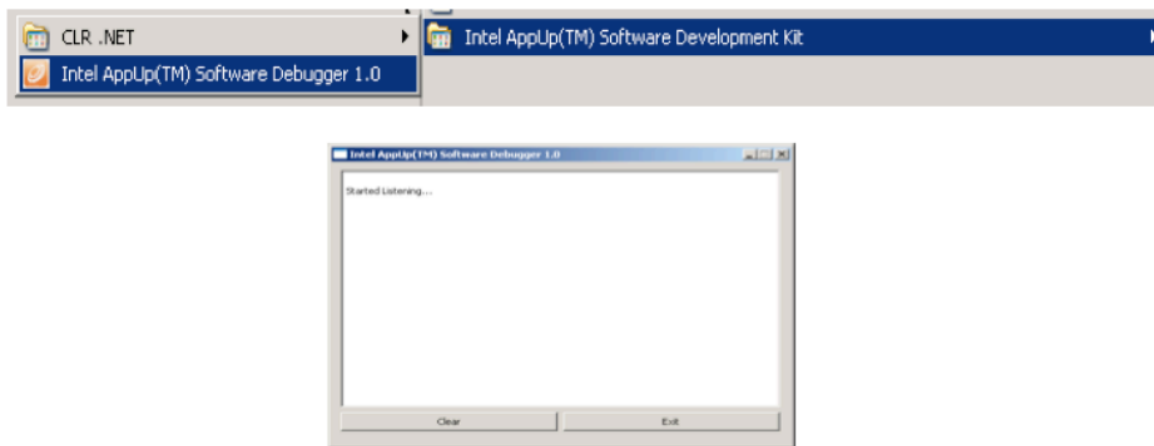
- Microsoft's runtime libraries (provided with the package).
- Python 2.X (<http://www.python.org>)
- A development environment that uses Python (recommended, but not obligatory).

### 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 "Application Test and Debug Service" (ATDS), a command line 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.

**Windows:** This ATDS is now the Intel AppUp Software Debugger and is run by calling it from the Start Menu and going to the Intel SDK program folder.



**Linux:** On the Terminal, run the ATDS with the command "run\_atds.sh" and stop it with the command "stop\_atds.sh". They are usually located in "/usr/bin".

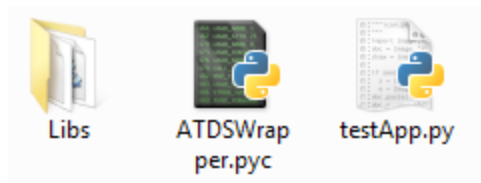
### Developing a Simple Console Application:

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, it can be project in any IDE or simply write in on any text editor.

Start by creating a python file that will serve as the application. In this case “testApp.py”. Then copy the file “ATDSWrapper.pyc” and the “Libs” folder to the same location.



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

```
from ATDSWrapper import *
```

We instantiate the wrapper class.

```
test = IntelAble()
```

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

```
connection = test.ConnectToIntel(0x11111111, 0x11111111, 0x11111111, 0x11111111,  
                                "11111111-1111-1111-1111-111111111111")
```

We need to check whether or not the application is authorized and connected

```
if connection == 0:  
    print "Connection Successful"  
else:  
    print test.GetErrorMessage(connection)
```

The function “test.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.

```
status = test.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 = test.BeginIntelEvent()
```

And

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

And to disconnect, the following function is executed

```
test.DisconnectFromIntel()
```

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:

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

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

### **Deployment of the Application:**

Once the code is complete, start the ATDS (Debugger in Windows) in the development environment and run the application with the following command:

```
python <Full Path to the python application>testApp.py
```

For example, on Windows, type:

```
python d:\Development\Python\testApp.py
```

**Note:** On Windows, you can also run the application by just double clicking on the Python file “testApp.py”. Linux users should use the command.

## List of Commands:

### 1. Connect to the App store:

*ConnectToIntel(param1, param2, param3, param4, baKnoGUID)*

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:

```
connection = test.ConnectToIntel(0x11111111, 0x11111111, 0x11111111, 0x11111111,  
                                "11111111-1111-1111-1111-111111111111")
```

#### Return values:

- **-10:** The GUID provided by baKno is invalid.
- **-5:** The library file "baKno.dll"(Win) or "baKno.so"(Linux) was not found.
- **-1:** The Application presented an Unknown Error.
- **0:** The Application started successfully.
- **1:** The application could not start.
- **2:** The Intel's Client Agent is not Available.
- **6:** The Application is not authorized.
- **7:** The Application ID has expired.
- **9:** The connection with the Client Agent reached a timeout.

## 2. Check the Application Status:

*GetIntelADPStatus()*

Command that checks the current status of the Application

### Return Values:

- **5:** The Application is authorized
- **6:** The Application is unauthorized.
- **7:** The Application ID has expired.
- **9:** The connection with the Client Agent reached a timeout

## 3. Save the Application Runtime (Optional)

*BeginIntelEvent()*

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

*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:

- **-1:** The Application presented an unknown error.
- **0:** The command has been executed successfully.
- **6:** The Application is not authorized.
- **7:** The Application ID has expired
- **8:** The "begin event" has not been called.

- **9:** The connection with the Client Agent reached a timeout.

#### **4. End the Connection:**

*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**

*Upgrade(param1, param2, param3, param4)*

This function calls the upgrade to the next application. The four string parameters represent the new GUID to call. I.e.: 0x22222222, 0x33333333, 0x44444444 and 0x55555555

##### **Return values:**

- **-1:** The Application presented an Unknown Error.
- **0:** The process started successfully.
- **1:** The process could not start.
- **6:** The Application is not authorized.