



# AppAble External for LiveCode

## Reference Guide

### Introduction:

baKno's AppAble External allows LiveCode developers to make desktop applications that connect to Intel's application store using a simple, yet complete set of functions.

### Package Contents:

- The Software License Agreement (License.txt)
- The AppAble External files (Externals)
- The dependency's libraries (Microsoft.VC90.CRT)
- A sample LiveCode stack that connects to the store
- A guide to create the MSI file for submission

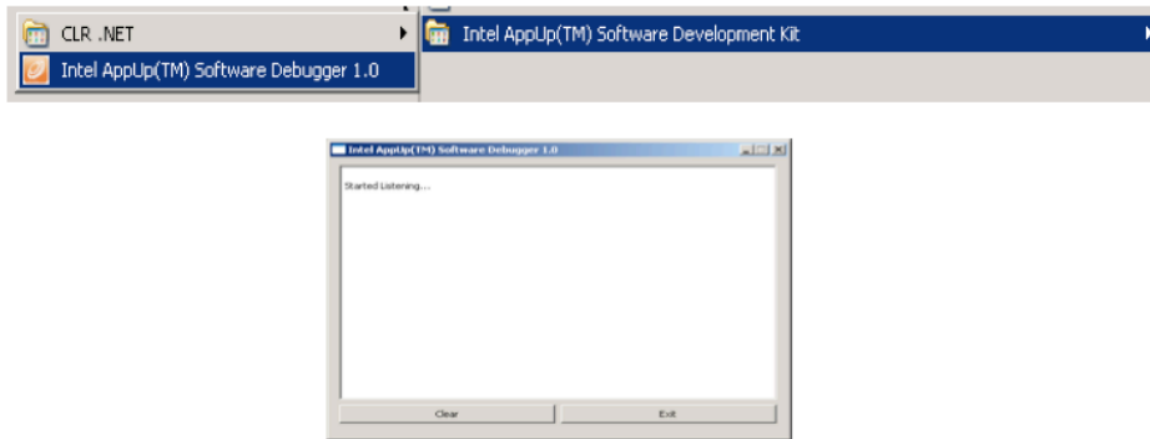
### Developer requirements:

- 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).
- LiveCode 4.x or later running on Windows.

### 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.



### **Add the Extension to LiveCode**

At the moment this external only works on Windows. Inside the AppAble package you will find a directory called "External". Place the contents of that directory inside the Externals directory at:

My Documents/My LiveCode/Runtime/Windows/x86-32/Externals

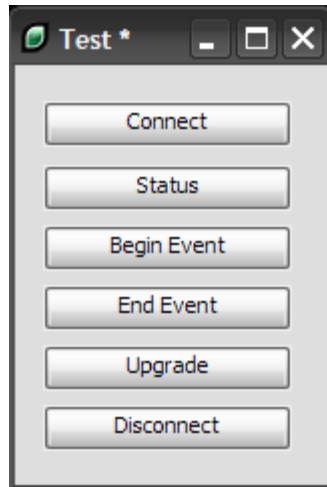
After the files are placed correctly, start LiveCode so the External can be loaded automatically.

### **Developing a Simple Standalone Application**

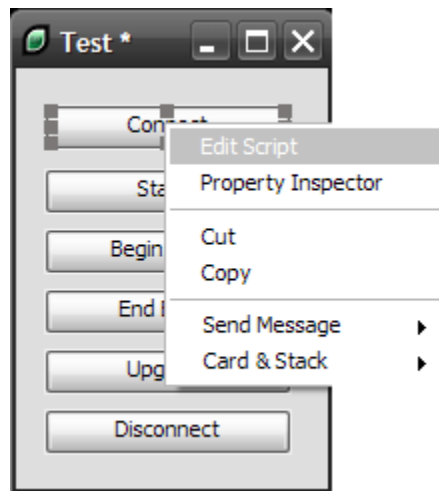
The example application does the following steps:

- Connects to the Client Agent.
- Checks the Authorization status.
- Begins and Ends a Recording event.
- Upgrades to another application.
- Disconnects from the Client Agent.
- Displays all messages pop-Up windows.

Start by creating a new, empty main-stack and add six push buttons named *“Connect”*, *“Status”*, *“Begin Event”*, *“End Event”*, *“Upgrade”* and *“Disconnect”*. These buttons will perform all main functions to interact with the Client Agent.



The next step is to add a script to each button, this is done by right-clicking on the button and select *“Edit Script”* from the pop-up menu.



For the Connect button, add the following script:

```
on mouseUp
try
    answer executeIntelApiCommand("Connect", "0x11111111", "0x11111111", "0x11111111",
        "0x11111111", "11111111-1111-1111-1111-111111111111")
catch theError
    answer "Error" && theError
end try
end mouseUp
```

The “Connect” parameter of the command command allows the application to connect to Intel's Client Agent. The next four parameters compose the GUID that Intel provides the developer to submit the application to the store. The last one is the ID provided by baKno to validate the external in a production environment.

**Note:** Check later in this document for more details of this function.

For the Status button, add the following script:

```
on mouseUp
  try
    answer executeIntelApiCommand("GetStatus")
  catch theError
    answer "Error" && theError
  end try
end mouseUp
```

This function allows to check the application's current authorization status at any moment in runtime.

For the Disconnect button, this script disconnects the application from the Client Agent:

```
on mouseUp
  try
    answer executeIntelApiCommand("Disconnect")
  catch theError
    answer "Error" && theError
  end try
end mouseUp
```

Intel allows developer to record the runtime of the application and is done easily, with just two commands. To begin recording the command is


```
executeIntelApiCommand("BeginEvent")
```

And to stop the recording

```
executeIntelApiCommand("EndEvent")
```

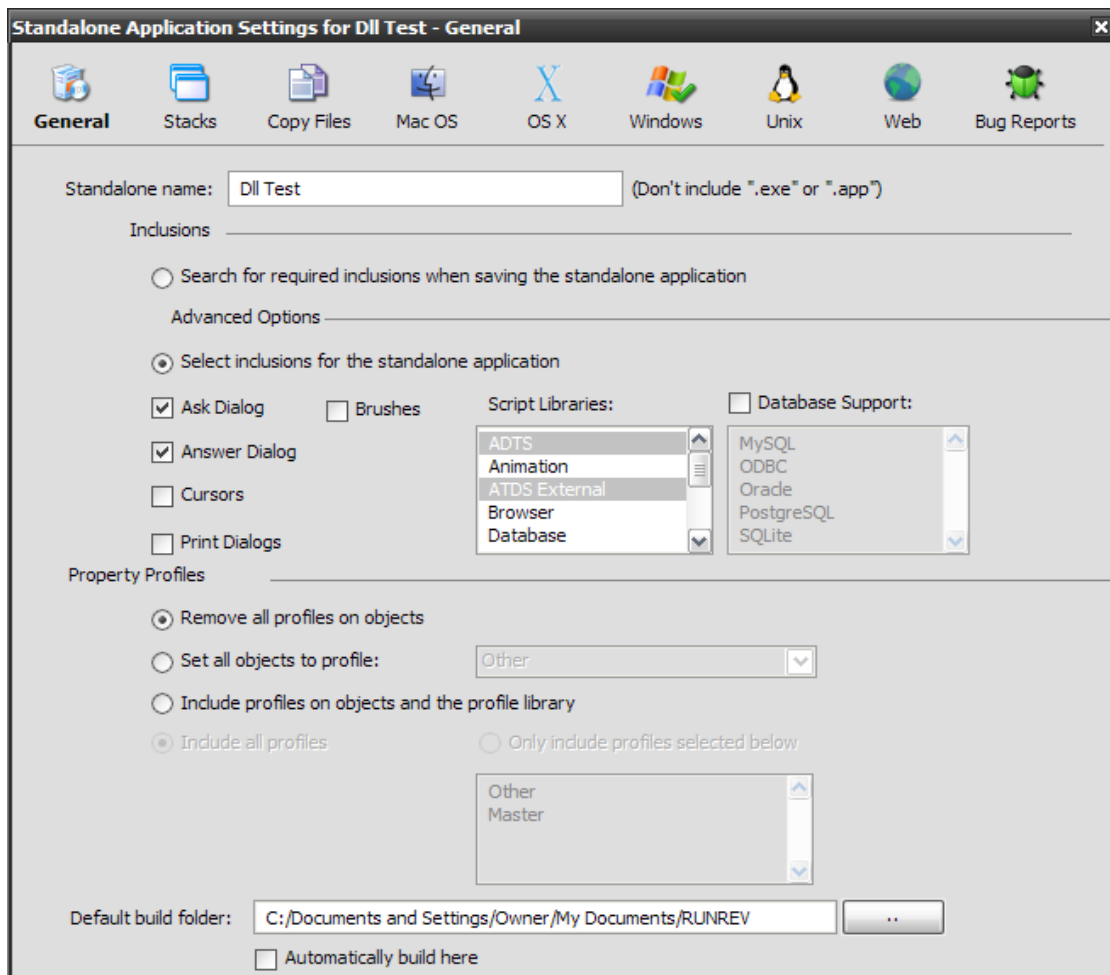
Intel also allows developer to upgrade the application to another with just one command, all that is needed is the GUID of the new application that Intel will provide. The command to use is the following:

```
executeIntelApiCommand("Upgrade", "0x22222222", "0x33333333", "0x44444444", "0x55555555")
```

To test the application, press on the  button on the “Tools” window to debug the simple application. Press the buttons from top to bottom and check the pop-up messages. If successful (LiveCode has loaded the external), a “SUCCESS” message should appear when the Connect button is pressed.

## Deployment of the Application

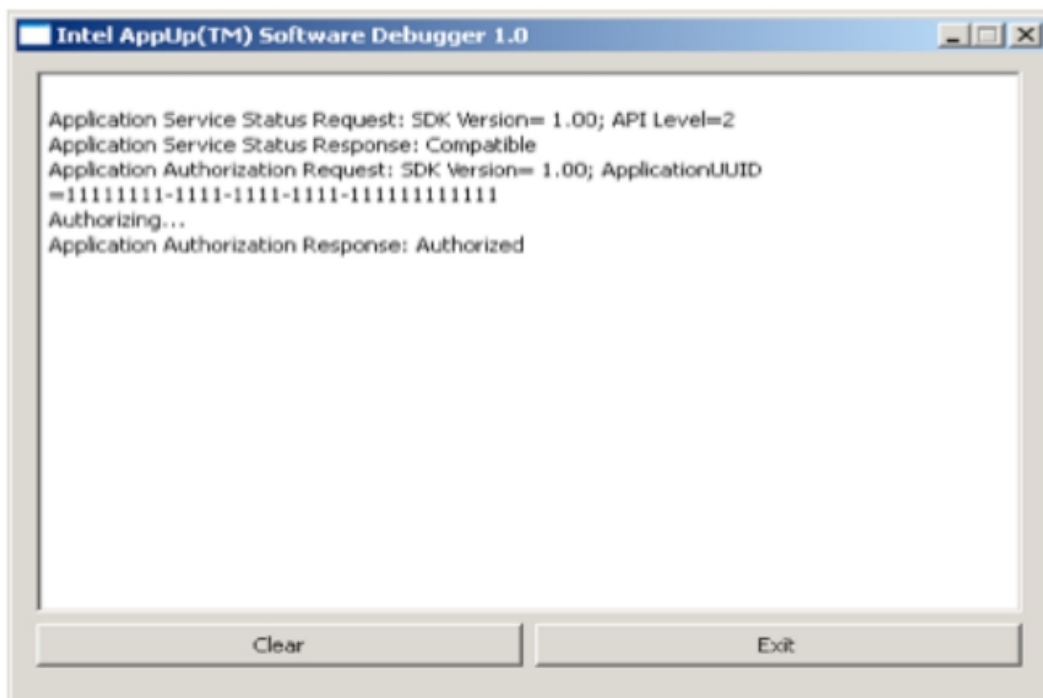
Once the tests are complete, then proceed to deploy the application by going to the “Standalone Application Settings” that is on the File menu in LiveCode in order to make sure the External is deployed. The following picture illustrates the recommended setup in the General tab (the only required).



In the picture, the option “Select Inclusions for the standalone application” is selected and the script libraries, ADTS and ADTS External, are selected.

Go to File->Save as Standalone Application and deploy the Windows test application. Move the folder of dependency's libraries (Microsoft.VC90.CRT) that is included in the package to the Externals folder next to the deployed application and all is ready to release.

To test for debugging, the Debugger from the SDK must start before the demo is executed.



## List of Commands

### 1. Connect to the App store:

*executeIntelApiCommand("Connect", string l1, string l2, string l3, string l4, string baKnoID)*

This command initializes the connection to the App store. The four "lx" parameters are part of the GUID that the Intel developer program provides to the developer.

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

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

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

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

*executeIntelApiCommand("Connect", "0x11111111", "0x11111111", "0x11111111", "0x11111111", "11111111-1111-1111-1111-111111111111")*

#### Error List:

- **Wrong parameters:** The parameters provided are invalid
- **Invalid Dll GUID: the dll** (not the application) GUID is invalid
- **All four parameters must be of numeric value:** Applies for Intel's GUID.
- **baKno.dll: File not Found:** The dll file could not be found
- **baKno.dll: Invalid File:** The dll file is invalid
- **NOT INITIALIZED:** The application could not start.
- **NO CLIENT AVAILABLE:** The Intel's Client Agent is not Available.
- **NOT AUTHORIZED:** The Application is not authorized.
- **ID HAS EXPIRED:** The Application ID has expired.

- **CONNECTION WITH THE CLIENT AGENT HAS TIMED OUT:** The connection with the Client Agent reached a timeout.

## 2. Check the Application Status:

*executeIntelApiCommand("GetStatus")*

Command that checks the current status of the Application

### Error List:

- **NOT AUTHORIZED:** The Application is unauthorized.
- **ID HAS EXPIRED:** The Application ID has expired.
- **CONNECTION WITH THE CLIENT AGENT HAS TIMED OUT:** The connection with the Client Agent reached a timeout.

## 3. Save The Application Runtime (Optional)

*executeIntelApiCommand("BeginEvent")*

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

*executeIntelApiCommand("EndEvent")*

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 *BeginEvent* is used, there must be *EndEvent* command later on.

### Return Values:

- **UNKNOWN:** The Application presented an unknown error.
- **NOT AUTHORIZED:** The Application is not authorized.
- **ID HAS EXPIRED:** The Application ID has expired



- **NO BEGIN EVENT HAS BEEN CALLED:** The “begin event” has not been called first.
- **CONNECTION WITH THE CLIENT AGENT HAS TIMED OUT:** The connection with the Client Agent reached a timeout.

#### 4. End the Connection:

*executeIntelApiCommand("Disconnect")*

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

**Note:** This command MUST be executed when the end uses closes the Application.

#### 5. Upgrade:

*executeIntelApiCommand("Upgrade", string l1, string l2, string l3, string l4)*

This function calls the upgrade to the next application. The four string parameters represent the new GUID to call. For debug purposes use 0x22222222, 0x33333333, 0x44444444, and 0x55555555 as parameters

##### **Return Values:**

- **UNKNOWN:** The Application presented an unknown error.
- **NOT AUTHORIZED:** The Application is not authorized.
- **ID HAS EXPIRED:** The Application ID has expired
- **CONNECTION WITH THE CLIENT AGENT HAS TIMED OUT:** The connection with the Client Agent reached a timeout.