AppAble Xtra for Director

Reference Guide

Introduction:

The baKno AppAble Xtra allows Director developers to make desktop applications that connect to Intel's app store on Windows platform.

Package Contents:

- The Software License Agreement (License.txt)
- The AppAble Xtra files (Xtra)
- A sample project that connects to the store (Sample)
- A text file with the scripts used in the sample.
- A guide to create the MSI file for submission

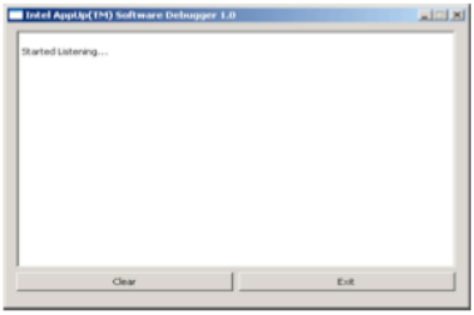
Developer requirements:

- The Intel's App store SDK. http://appdeveloper.intel.com/en-us/
- Macromedia / Adobe Director

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.

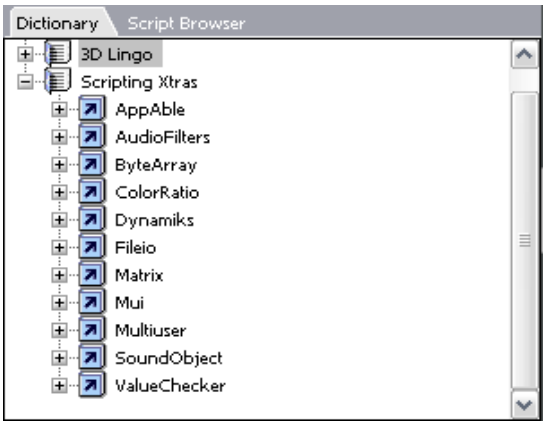


Adding the AppAble Xtra to Director

To add the Xtra simply copy the file “AppAble.x32” that comes with the package to the path where Director contains all Xtras. For example, in Director 11:

C:\Program Files\Adobe\Adobe Director 11\Configuration\Xtras

Reopen Director and the Xtra should be available in the Dictionary

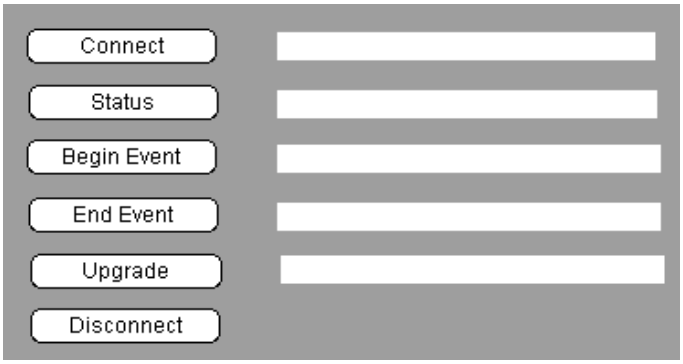


Developing a Simple Application

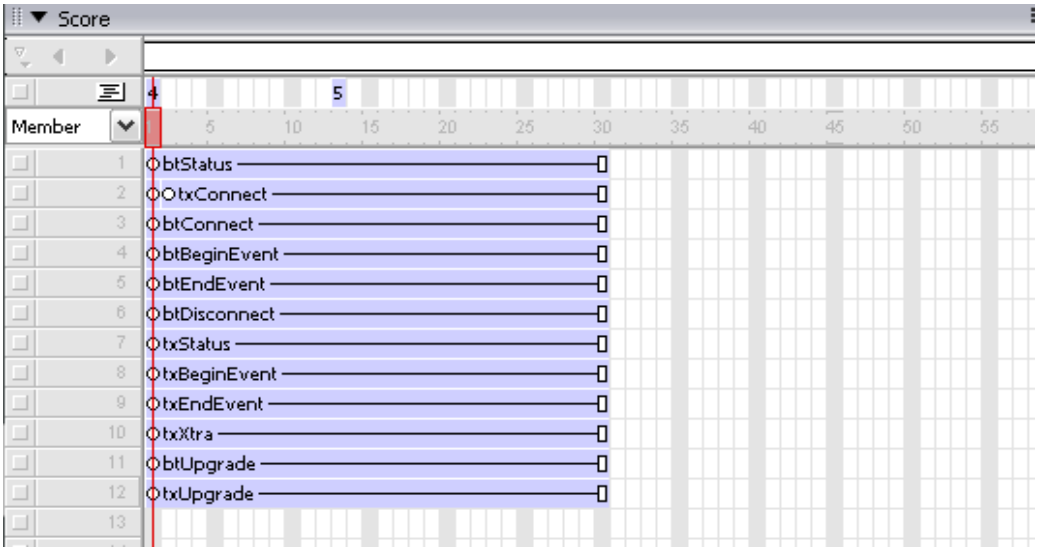
The example application does the following steps:

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

Prepare a small movie in Director where you add 5 push buttons and five fields that will make the GUI for our application.
The first field will warn if the Xtra is not avaialable on the application, name it "txXtra". Name each button and field for ease in development (i.e. btConnect, tx Connect, btStatus, etc.). The result should be like in the following image.



Make sure they all last the same amount of frames



The next step is to add a simple behavioral script to the movie. This will return to the first frame to make a simple endless loop

```
on exitFrame me
  go to frame 1
end
```

For each button add a simple behavior script that will call each handle that corresponds to a process. For example, for the Connect button we add the following lines

```
on mouseUp
  try_Connection()
end
```

Now create an Utility Script that will be general for the application, and start by declaring a global variable for the Xtra

```
global baKno
```

Declare the first handle that will connect to the AppStore. It will be called once the “Connect” button is pressed.

```
on try_Connection
```

We assign to the global variable the AppAble Xtra.

```
baKno = xtra "AppAble"
```

The variable inPath contains the path where the .exe application will be located

```
inPath = _player.applicationPath
```

If you wish to debug the application, use the full path where the file “baKno.dll” (included in the package) is located.

```
inPath = “C:\\AppAbleExample\\”
```

To connect to Intel's App Store we used the following call

```
result = connectToIntel(baKno, inPath,  "0x11111111", "0x11111111", "0x11111111",
"0x11111111", "11111111-1111-1111-1111-111111111111")
```

We add the path, the four pieces of Intel's GUID and the ID provided by baKno either for debug or production environments. The return integer value can be examined by the handle “*get_Message*” which is included in the script text file (Optional, if you wish to use another handle or just work with the value is fine)

```
message = get_Message(result)
put message into field "txConnect"
end
```

To check the status of the application once the application is connected we use the following handle

```
on check_Status
result = checkApplicationstatus(baKno)
message = get_Message(result)
put message into field "txStatus"
end
```

Sometimes is 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.

```
on begin_Event
result = beginIntelEvent(baKno)
message = get_Message(result)
put message into field "txBeginEvent"
end
```

And

```
on end_Event
  result = endIntelEvent(baKno)
  message = get_Message(result)
  put message into field "txEndEvent"
end
```

To disconnect, the following handle is executed

```
on disconnect
  result = disconnectFromIntel(baKno)
end
```

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:

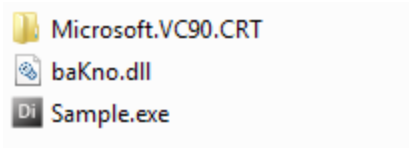
```
on upgrade
  result = upgrade(baKno, "0x22222222", "0x33333333", "0x44444444", "0x55555555")

  message = get_Message(result)
  put message into field "txUpgrade"
end
```

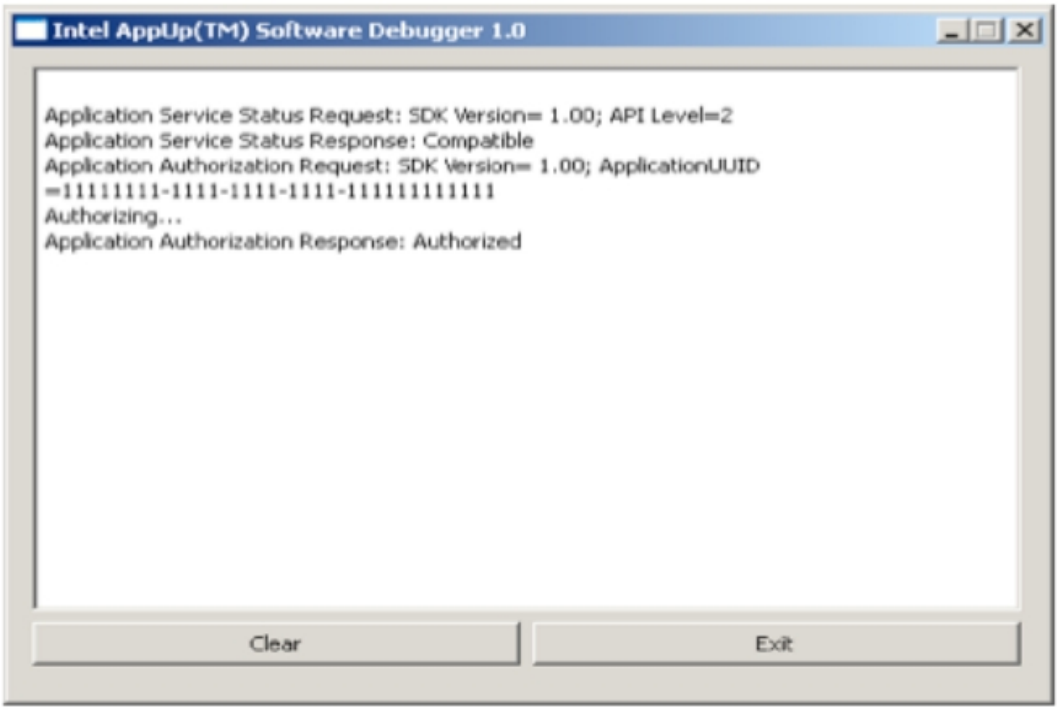
The full source code of the example is provided within this package.

Deployment of the Application

When the application is published as a destop Application (exe file), make sure thatthe dll file “baKno.dll” is located on the same folder where the application is



the Debugger from the SDK must start before the demo is executed.



List of Commands

1. Connect to the App store:

connectToIntel(object me, string path, string param1, string param2, string param3, string param4, string baKnoID)

This command initializes the connection to the App store. The first parameter is the AppAble Xtra and the second is the path to locate the dll file “baKno.dll”. Each of the next four 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:

result = connectToIntel(baKno, inPath, "0x11111111", "0x11111111", "0x11111111", "0x11111111", "11111111-1111-1111-1111-111111111111")

Return values:

- **-10:** The baKno ID is invalid.
- **-4:** One or more parameters are invalid.
- **-3:** Cannot locate the file "baKno.dll".
- **-2:** The dll file "baKno.dll" is invalid.
- **-1:** An unknown error has occurred
- **0:** The connection has been successful.
- **1:** The connection could not initialize.
- **2:** No client available.
- **3:** Incompatible version.
- **4:** The data is too big.
- **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:

checkApplicationstatus(object me)

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(object me)

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

endIntelEvent(object me)

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(object me)

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

5. Upgrade the Application

upgrade(object me, string path, string param1, string param2, string param3, string param4)

This function calls the upgrade to the next application. The four string parameters represent the new GUID to call.

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.
- **7:** The Application ID has expired.