



AppAble Plugin for Unity Pro

Reference Guide

Along with this documentation you will find a folder called “AtomSDK Sample Project” which is a Unity Pro project that includes all the necessary files, objects, scripts and structure. And another folder called “DLL Wrapper” which contains the actual Plugin, some other necessary files that make it work, and its documentation.

Introduction

Intel created an SDK to allow developers an easy integration of their current developments with the application store. This SDK contains several components and among them the most important ones are the actual libraries that the developer needs to integrate within his/her code to communicate with the store.

Unfortunately for Unity developers, these libraries are static .LIB and Unity only accepts dynamic .DLL ones. The solution proposed by a Unity engineer was to create a DLL-Wrapper, and this is what baKno developed and integrated into a Unity Project for you to publish your games into this new App Store.

First, you will need to sign into the Atom Development Program at the following Intel’s site:

<http://appdeveloper.intel.com/>

Download the SDK, read the documentation and get familiar with the program and the SDK in general.

When ready, using the same site, create a new application under your account. You will only need to enter the name and minimum information as a starting point. And if accepted, it will be listed under your account and a GUID code will be assigned. You will need this code later to build the final application.

When getting familiarized with the SDK, please note the location of two important files:



In the AppUp SDK 2.0 these files have been renamed “run.bat” and “stop.bat”

There are two environments: Debug and Production. The idea is to develop everything under Debug and only when everything works, change to Production to build the final application and submit it to the App 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 App Store. By running the Debug application you can check if all communication stages work as described in the docs. This ATDS is run and stop with the two files mentioned before.

Sample Project

The sample project contains two scenes: IntroScene and SceneX.

IntroScene is the one that is loaded first when the game starts. This scene has two different objects: IntroObject and IntelObject.

IntroObject has a script called Intro. This script is the one that checks the initial Communications with the App Store and validates the Application. This is why it has to be run just once when the application starts.

IntelObject is a GUIText object that shows the error messages and therefore it has a C# script called “baKno.cs” that communicates with the Plugin. This object has to be located in all scenes that you want to check status against the App Store.

SceneX is just a sample scene and it just contains a different version of the IntelObject. This version has an extra script called “Appstore.js”. We recommend to place the IntelObject as it is in the SceneX (copy and paste) in all your scenes, but if you just want to put it into the main game scene, that is fine also.

Understanding this basic structure should be enough for you to decide how to place these game objects into your own Project.

Please remember to copy the Plugins folder as it is into your Project as well. If you are developing in Mac OS and try to play, it will show an error because the plugin is made to run on Windows computers.

To test your Project you will need to create a Windows build, make sure the whole Plugins folder is exactly the same as the one on the Project (usually it does not copy

all the files) and test it on a Windows computer, a Netbook preferably. But before running the game please remembre to run the runATDS.bat file mentioned before.

If it runs fine, and you are ready to build the final version, please go to your Unity environment, open the script called "baKno.cs" and replace the portion where it has the debug GUID (0x11111111, 0x11111111, 0x11111111, 0x11111111) with the one that Intel assigned to your application on their website.

DLL Wrapper

This folder contains the same files located on the Plugins folder mentioned before. Additionally it has its own documentation for you to better understand its functionality.

If you prefer to implement your own way to communicate and validate your application with the App Store, or if you want to better understand how the sample Project does it, please refer to this documentation.

The documentation contains the list of commands implemented by the DLL Wrapper. All these commands correspond to the same functionality offered by the SDK libraries.