AppAble Library For Delphi

Reference Guide

Introduction:

The AppAble library allows Delphi developers to make applications on their IDE of choice on the Windows platform 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 Delphi project that access the library (Sample)
- A guide to create the MSI file for submission

Note: The Delphi file “IntelAble.pas” 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.

The IntelAble.pas contains the Delphi functions and procedures that are implemented in the IntelAble.dll. Those functions provide interaction with the Intel AppUp runtime. The library works with Delphi 6 or higher.

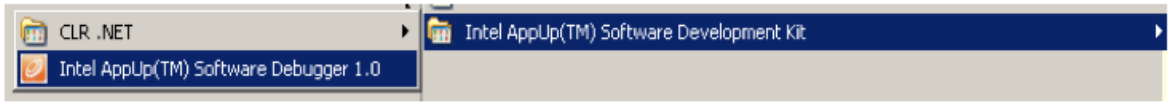
Developer requirements:

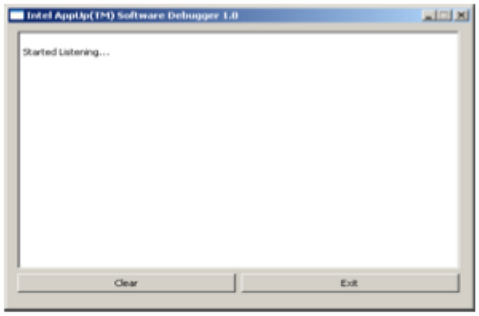
- The Intel's App store SDK. It can be obtained here: http://appdeveloper.intel.com/en-us/
- For Delphi development download Intel SDK for .NET.
- Microsoft's runtime libraries (provided with the package).
- Delphi Development Environment.

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.





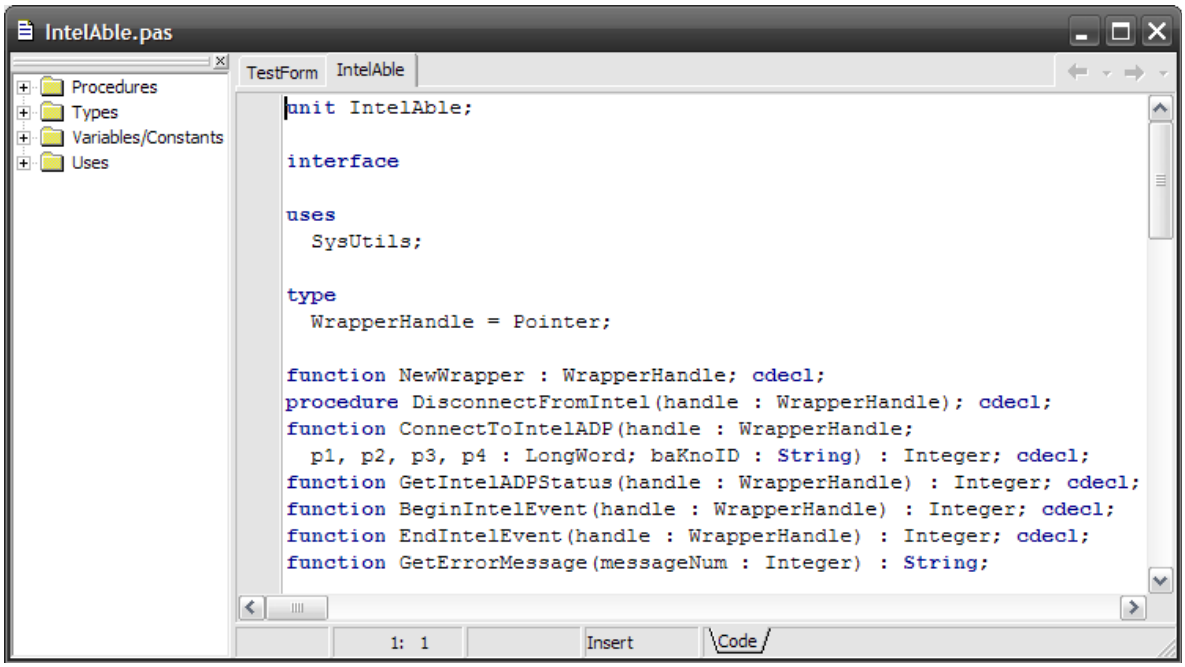
Developing a Simple Application:

The example application does the following steps:

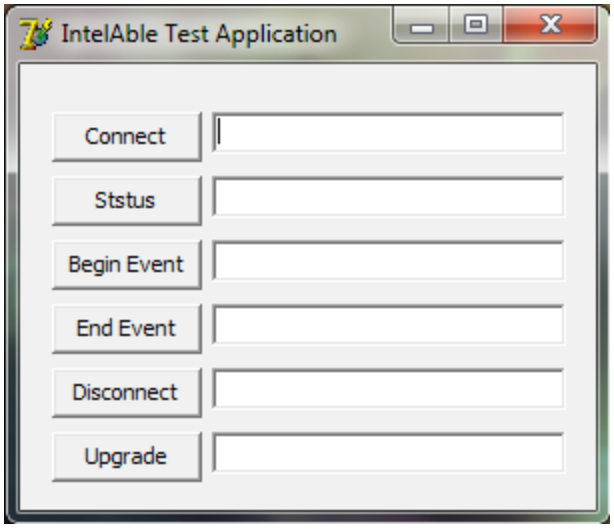
- Connects to the Client Agent
- Checks the Authorization status
- Begins an event
- Ends the event
- Disconnects from the Client Agent

First we need to create a simple Windows project, it can be done with the default environment when Delphi starts (Delphi 6.0 or later versions). If there is no new project, create one with the default settings. Then save the project in the folder of choice.

Next, copy all the library files and dependencies folder that are included in the package on the same folder as the application project. Include the file “IntelAble.pas” to the project in order to access the functions that connect to the App Store.



Go to the form and add 6 buttons with a text field next to each one of them



Now go the the form's code (interface) and add the AppAble's Interface to the list

```
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, IntelAble, StdCtrls;
```

Proceed to declare two more variables, an integer that will contain the reply message from the library and a handle that access the wrapper

```
var
  Form1: TForm1;
  mess: Integer;
  Wrapper: WrapperHandle;
```

Proceed to add the Click event go for the first button and begin the connection to the App Store

```
Wrapper := NewWrapper;
mess := ConnectToIntelADP(Wrapper, $11111111, $11111111,
        $11111111, $11111111, '11111111-1111-1111-1111-111111111111');
```

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

```
if mess = 0
 then Edit1.Text := 'Connection Successful'
 else Edit1.Text := GetErrorMessage(mess);
```

Note: The Function “GetErrorMessage” translates the result the library provides into a readable string message for the end user

Create the Click event for the second button to check for the application status

```
mess := GetIntelADPStatus(Wrapper);

if mess = 5
then Edit2.Text := 'The Application is Authorized'
else Edit2.Text := GetErrorMessage(mess);
```

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 (use them on separate button events).

```
mess := BeginIntelEvent(Wrapper);
```

And

```
mess := EndIntelEvent(Wrapper);
```

To disconnect, the following proceedure is executed

```
DisconnectFromIntel(Wrapper);
```

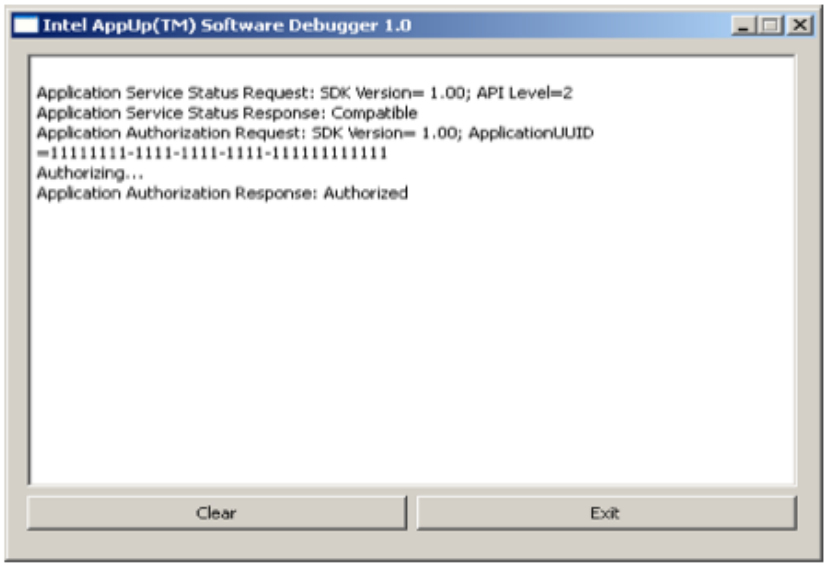
The full source code of the example is provided within this package.

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

```
mess := Upgrade(Wrapper, $22222222, $33333333, $44444444, $55555555);
```

Deployment of the Application:

Once the code is complete, build the entire project. Then, start the Debugger in the development environment and run the application on the IDE for debbuging.



List of Commands:

1. **Connect to the App store:**

   function *ConnectToIntelADP(handle : WrapperHandle;*
   p1, p2, p3, p4 : ***LongWord****; baKnoID :* ***String****) :* ***Integer;***

   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:

   mess := ConnectToIntelADP(Wrapper, $11111111, $11111111,
   $11111111, $11111111, '11111111-1111-1111-1111-111111111111');

   Return values:

   - **-10:** The GUID provided by baKno is invalid.

- **-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:**

   function *GetIntelADPStatus(handle : WrapperHandle) :* ***Integer***

   Command that checks the current status of the Application

   Return Values:

   - **2:** The Intel's Client Agent is not Available.
   - **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)**

   function *BeginIntelEvent(handle : WrapperHandle) :* ***Integer***

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

   function *EndIntelEvent(handle : WrapperHandle) :* ***Integer***

   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.
- **2:** The Intel's Client Agent is not Available.
- **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:**

procedure *DisconnectFromIntel(handle : WrapperHandle)*

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

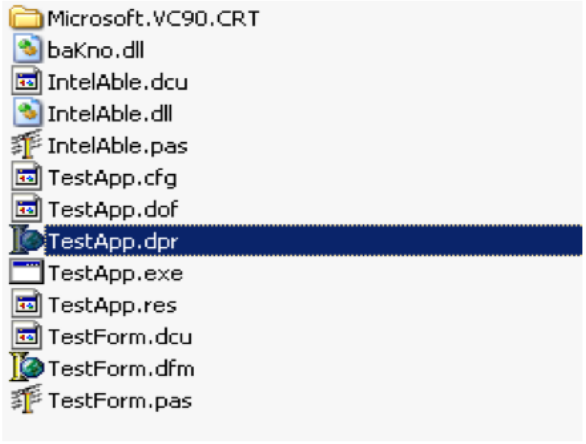
Note: This command MUST be executed when the end uses closes the Application. If it's executed earlier, the application may close by itself immediately. This is due that the App store has a record of the application that is running and terminates the process automatically.

5. **Upgrade the Application**

function *Upgrade(handle : WrapperHandle; p1, p2, p3, p4 :* ***Cardinal****) :* ***Integer;***
Function that performs upgrade or get another app from the store.

Note: This command MUST be executed after the application has registered to the AppUp when it's loaded. The four parameters are the GUID of the next application.

Application code for the simple application is located in the baKno distribution under 'DelphiLibrary.zip\AppAble Library\Samples\Delphi 7 and over'.



For Delphi 6 you must remove all *.dcu and recompile the entire project.