F O R U M   N O K I A

User Guide for Hybrid Application Generator

Version 1.0; June 24, 2010

NOKIA

Contents

Change history

June 24, 2010	Version 1.0	Initial document release

1 What is Hybrid Application Generator?

Hybrid Application Generator (or simply 'Generator' from now on), is a framework that allows developers to create native Qt applications that show HTML-based content, including applications that use JavaScript™ and Cascading Style Sheets (CSS). The framework serves as an interface between JavaScript functionality (such as that found in Web Runtime [WRT] widgets) and Qt APIs. For example, when your program requests the application to switch to full screen mode, the framework connects the JavaScript function call to relevant native platform functionality.

The result is a system that allows developers to use web technologies to quickly create services that can then be converted to Qt applications to enable simple delivery and installation to a variety of mobile devices. However, unlike pure widget applications, Hybrid Applications retain the power and flexibility of native code.

Hybrid Application Generator is only available for Windows at this point.

2 Installation

Install Hybrid Application Generator by running the installer executable. There are no limitations for the installation directory location. The core application is required, but you can choose not to install start menu icons or demo applications.

The installer creates an uninstaller that removes the Hybrid Application Generator installation. The uninstaller can be run from the start menu folder or through Windows Add/Remove program functionality.

Hybrid applications can be generated locally using a suitable SDK, or remotely with Remote Compiler.

2.1 Installing without SDK

The Generator can be used in stand-alone mode without a Qt SDK. In that case the Hybrid Application Generator will use Nokia Qt SDK Remote Compiler to create installable packages for

2.2 Installing the SDK for local application generation

The Generator supports both Nokia Qt SDK (release candidate and later versions) and full Symbian S60 SDK. Unless you require Symbian native code compilation, we suggest you to download and install only the Nokia Qt SDK. Nokia Qt SDK is available from http://www.forum.nokia.com/Develop/Qt/Tools/.

Note: only the Symbian platform is supported when using a local SDK. The produced project can be transferred to Scratchbox for Maemo compilation, but instructions for this are out of the scope of this document.

2.2.1 Optional Symbian S60 SDK

See Symbian S60 SDK installation instructions at http://developer.symbian.org/wiki/index.php/Qt_Quick_Start to install the full Symbian C++ development environment. The required phases are 'Symbian C++ Development Environment' and 'Patch Symbian C++ Development Environment'. You will have to install Java, Perl, ADT, SDK and patches for Open C and Open C++.

2.2.2 Required Qt version

If you are using the Nokia Qt SDK or the remote compiler, no additional Qt libraries are required. Hybrid applications will compile both for Qt 4.6.3 and Qt 4.6.2.

If you are planning to use the full Symbian SDK or Scratchbox, note that Qt Mobility is required, too. The latest files for Mobility Project can be found at http://qt.nokia.com/products/appdev/add-on-products/catalog/4/new-qt-apis/mobility, or directly from the Scratchbox repository with a `qtm-` prefix. Installing the mobility API (headers and necessary binaries) to your Symbian environment is done by uncompressing the "source and symbian binary" .zip, and among the uncompressed files, locating another zip, like "qt-mobility-1.0.1-epoc32-5.0" zip, and uncompressing it in the root folder of your SDK. Make sure to pick the correct zip matching your SDK to uncompress. TODO – check this paragraph / review.

```
TODO: there are still problems with 4.6.1. ???
```

```
TODO: On vista, use …\vistapatch\Env.BAT ???
```

When first running the Hybrid Application Generator, the application will try to find paths to required Qt directories. Make sure that the correct path to Qt's `qtenv2.bat` (for Nokia Qt SDK) or `qtenv.bat` (when using a normal Qt SDK and the the full Symbian S60 SDK). For example, a correct path to `qtenv2.bat` may look like `C:\NokiaQtSDK\Symbian\SDK\bin\qtenv2.bat`.

3 Generating applications

Hybrid Application Generator contains a generator application that can be used to create hybrid applications from WRT widgets, directories containing HTML content, or directly grabbing web pages. The application is launched by double-clicking `HAG.exe` in the application directory, or by selecting the application shortcut from the Start menu.

3.1 Configuring the build target

From the build menu, select the target platform. You can target Symbian devices if you have a local SDK installed, and both Symbian and Maemo devices when using the remote compiler. Some of the Generator settings are platform specific, so remember to verify the settings dialog contents always before launching a new build process. Take extra care with the plug-in selection: plug-in settings are not persisted and will thus always reset to default settings when switching targets or restarting the application.

3.1.1 Remote compiler

To use the remote compiler, you will need a Forum Nokia account. You can join the Forum Nokia at http://www.forum.nokia.com/.

From the build menu, select Connect to Remote Compiler. You will need to give your Forum Nokia account username and any applicable proxy settings. Your Forum Nokia password is requested after clicking Sign In. You can select several different remote compiler targets after successful login.

When you have selected a suitable target and checked the settings, choose Build Project from the Build menu. The Generator will upload your project to the remote compiler and will download the resulting installer if the compilation completes successfully.

The remote compiler uses Scratchbox as its compilation environment when compiling for Maemo. Hybrid plug-ins are not currently supported in Maemo.

3.1.2 Local compilation

Local compilation will be available if you have installed a local SDK as mentioned in Chapter 2. Simply confirm the settings and select Build Project from the Build menu. Due to restrictions in MADDE environment, only the Symbian platform can be targeted with local SDKs.

3.2 From WRT widgets

To create a hybrid application from a WRT widget, choose Open Widget File from the File menu and select a widget file (WRT widgets usually have a `.wgz` extension).

Using Preview from the Build menu, you can have a tentative look at how the widget behaves in a mobile environment, including the ability to choose between different screen resolutions. Note that there are differences from the PC environment – for example, the preview window does not emulate Symbian soft keys, and the menu items are located in the application menu bar.

From the build menu, selecting Build Project will start the conversion. Once the build finishes, the resulting installer (`application.sis`) can be found in the project folder. The folder can be opened from the main window using Open Build Directory.

3.3 From HTML directories

If you have an HTML/JavaScript application, it hardly makes sense to create a WRT widget of it when planning to create a hybrid application. Thus, by selecting Open Directory from the File menu, you can directly convert directories to Qt applications. Once a directory is selected, the usage is similar to the conversion of WRT widgets.

The selected folder may have subfolders, but your main HTML file has to reside in the exact folder you selected. The generator will look for `index.html` and `main.html` files, and if neither is found, will use any other HTML file available. This may result in using a nondesired file as the starting point – use the preview functionality to check if the outcome is as desired.

3.4 Grabbing web pages

The generator has a web grabber, too, which you can use to fetch your HTML files directly from the server. Choose Open URL from the File menu, and give an address to be used. The generator will fetch the site to a temporary folder, and you can then use the generator normally – you can preview the just-fetched site and build an installable Qt application out of it.

3.5 Installing hybrid applications

After acquiring an installation package (`.sis` for Symbian, or `.deb` for Maemo) you should try that the application installs and runs successfully.

When using Symbian, connect the device to the computer with an USB cable in PC Suite mode and double-click the installation file (or select Install SIS... in the Generator main window). Make sure that the device contains the same version of the Qt libraries and the latest Qt Mobility package.

When using Maemo, first make sure that you have installed the `rootsh` package via the Application manager. Connect the device to the computer with an USB cable in PC Suite mode. Copy the `.deb` file to the Documents folder. Open the X Terminal and type the following commands:

```
sudo gainroot
cd MyDocs/.documents/
dpkg -i <application>.deb
```

Note that currently only one hybrid application can be simultaneously installed. The restriction comes from included DLL and library files: each application contains the framework DLL and possibly same hybrid plug-ins. Symbian installer will refuse to overwrite existing files owned by other applications, and the installation is thus halted. At this point, you may want to changing the DLL and library names if you are planning to distribute your hybrid application.

4 Using the framework

If you wish to develop something besides basic web page functionality, you can call the Hybrid framework interfaces. The default distribution allows you to add and remove items in the application menu, show or hide soft keys, open external applications or URLs, change display orientation, and more. The distribution also contains plug-ins that enable access to location, photo, and sensor APIs, thus allowing you to access the global positioning system (GPS), motion sensor, and images stored to the mobile device memory.

4.1 Using the framework without the Generator

The Generator application is a convenience tool for creating hybrid projects, but there is no requirement for using it. The application installation folder contains templates for both the application itself and for the Hybrid framework that enables the mixing of web content and Qt code, and you are welcome to use the code in a way that best suits you.

- The Hybrid framework is in the folder `fw/`
- Plugins, both pre-compiled and in source form, are in the folder `mw/`
- Application and package templates are in the folder `templates/`

The template and framework files contain tags that the Generator application fills at the beginning of each compilation according to project details and settings. It may be beneficial to create a project with preferred settings, build it either with a local or the remote compiler, and use the resulting source folder as the basis of your custom hybrid application.

It is currently not possible to bring the changes back to a generated project if you customise the project or framework files. When building the application again from within the Generator, the template files will be copied over any existing files. In such cases it is best to open the project in Qt Creator which is installed with the Nokia Qt SDK. New versions of Qt Creator include support for remote compilation, too.

5 Examples

5.1 Demo applications

The application contains demo applications, both WRT widgets and simple web pages. The widgets can be used as shown in Section 3.2, 'From WRT widgets', and the web page preview/build process is explained in Section 3.3, 'From HTML directories'. In general, widgets should have panning disabled, and web pages benefit if panning is enabled. Panning settings are found in the Settings dialog.

- BreakDown widget. BreakDown is a simple Breakout clone in JavaScript. The game uses motion sensors to control the paddle. Thus, you won't be able to preview this demo in your workstation – instead, you should make a Symbian build and install it to your device. The BreakDown demo works only with resolution 360x640 (for example, N97 in portrait mode).

- MeeGo.com web page. This is a grab of the MeeGo.com site showing MeeGo architecture. The purpose of the demo is to show an example of a hybrid application made out of a web page. The page still allows normal navigation via hyperlinks.
- Menu example web page. This is the example shown in Section 5.2, 'Adding menu items to the application menu'.
- GPS example web page. This is the example shown in Section 5.3, 'Getting the current GPS position'.

5.2 Adding menu items to the application menu

The following code snippet will add two menu items to the application's menu, accessible via the left soft key. In preview mode, the items will be shown in the window's menu bar.

```
<script type="text/javascript">

function handler(id) {
   switch (id) {
      case 100:
      case 101:
         alert(id);
         break;
      default:
         alert("Unknown method!");
   }
}

var menu = window.menu;
var m1 = new MenuItem("Item one", 100);
var m2 = new MenuItem("Item two", 101);

m1.onSelect = handler;
m2.onSelect = handler;

menu.append(m1);
menu.append(m2);
menu.getMenuItemById(101).setDimmed(true);

</script>
```

5.3 Getting the current GPS location

The code below will take the current GPS location and show an alert box containing the device's latitude, longitude, and altitude. Because the generator preview obviously doesn't have access to location services, running the preview function with this example will return a fixed location in Barcelona. Furthermore, the location service may return invalid data until it gets a GPS fix, so data validity should be tested. See the GPS demo for one possible implementation.

```
<script type="text/javascript">

function callback(id, code, result) {
   var latitude = result.ReturnValue.Latitude.toString();
   var longitude = result.ReturnValue.Longitude.toString();
   var altitude = result.ReturnValue.Altitude.toString();
   alert(latitude + "\n" + longitude + "\n" + altitude);
}

var updateObj = new Object();
updateObj.PartialUpdates = false;

var criteriaObj = new Object();
criteriaObj.LocationInformationClass = "BasicLocationInformation";
criteriaObj.Updateoptions = updateObj;
```

```
var locationObj = device.getServiceObject("Service.Location",
"ILocation");
var result = locationObj.ILocation.GetLocation(criteriaObj, callback);

</script>
```

6 Documentation

The Hybrid framework replicates parts of the WRT API. The easiest way to access the functionality of the framework is to include the file `hybridapi.js` to your project and use the methods it provides. You can, of course, bypass `hybridapi.js` completely and use the framework's interfaces directly, in a manner similar to as how `hybridapi.js` operates.

The following collection of Forum Nokia WRT documents presents features currently supported by the Hybrid framework. The generator automatically includes `hybridapi.js` to your application, so you can use these examples without any additional steps. In other words, even if you implement some features using `hybridapi.js`, you do not have to include it to the project yourself since it is considered to be part of the Hybrid framework.

JavaScript menu object:

http://library.forum.nokia.com/topic/Web_Developers_Library/GUID-857CF71D-3398-40C2-981A-DEC428A7BA6B.html

JavaScript widget object:

http://library.forum.nokia.com/topic/Web_Developers_Library/GUID-6CD2776F-A868-4280-967F-4EB426212556.html

JavaScript MenuItem object:

http://library.forum.nokia.com/topic/Web_Developers_Library/GUID-111DE423-9C84-4E4B-A45E-15081FE2A30D.html

JavaScript device object:

http://library.forum.nokia.com/topic/Web_Developers_Library/GUID-571AF37F-1E95-462A-92D2-FEA2E62F1559.html

Note: For the device object, only `Service.Location` and `Service.Sensor` are currently supported.

Platform services cannot be used via the Hybrid framework.

7 Hybrid plug-ins

A Hybrid plug-in is an extension to the Hybrid framework. With a Hybrid plug-in you can extend the framework to support new native APIs. For example, if you have an HTML/JavaScript widget that requires authentication, you can implement an authentication plug-in using C++ and expose the API automatically to your JavaScript code.

Hybrid plug-ins are currently supported only in the Symbian platform. Maemo support is planned for future releases.

7.1 Implementing a plug-in

Hybrid framework plug-ins are based on Qt plug-in architecture. See the Qt plug-in ‘How To’ at http://doc.trolltech.com/4.6/plugins-howto.html.

Each plug-in needs to implement the following interface:

```
#ifndef HYBRIDPLUGININTERFACE_H
#define HYBRIDPLUGININTERFACE_H

#include <QMainWindow>

class QWebFrame;

class HybridPluginInterface
{
public:
    virtual ~HybridPluginInterface() {}

    /**
     * Set environment variables.
     *
     * @param parentWindow Parent window, can be used to control
     *        menu bar for example from plugin.
     * @param webFrame Webframe where the plugin was loaded into.
     */
    virtual void setEnvironment(QMainWindow *parentWindow,
                                QWebFrame *webFrame)=0;

    /**
     * Return JavaScript object name. Used to identify
     * this plugin.
     *
     * @return JavaScript object name.
     */
    virtual QString jsObjectName()=0;

    /**
     * Return instance of the API implementation.
     *
     * @return Instance of the API implementation.
     */
    virtual QObject *jsObjectInstance()=0;

    /**
     *
     * @return Name of the javascript wrapper file.
     */
    virtual QString jsFileName()=0;
};

QT_BEGIN_NAMESPACE
Q_DECLARE_INTERFACE(HybridPluginInterface,
com.nokia.fwui.HybridPluginInterface/1.0");
QT_END_NAMESPACE

#endif // HYBRIDPLUGININTERFACE_H
```

All of the slots from the object returned by the `jsObjectInstance()` method are exposed in JavaScript code.

7.1.1 Plug-in structure

Hybrid Application Generator will build all the selected plug-ins when the application is generated. In order for the Generator to find your plug-in, you must place it under the `mw/` directory. It should be a normal Qt plug-in implementing the `HybridPluginInterface` interface.

A trivial example plug-in is included with the application in directory `templates/plugins/example`. When implementing a new plug-in, you may want to consider to use the example and extend it to suit your needs. The example plug-in is fully functional and can be used in project without changes simply by copying it to the `mw/` directory.

Below is description of the example plug-in files. Compare the contents of `example.h` and `example.js` files to see how the Qt slots can be accessed.

Plug-in description for generator (parsed for 'Plugins' tab in Settings dialog)
```
    templates\plugins\example\example.xml
```

Project file
```
    templates\plugins\example\example.pro
```

Hybrid plugin interface implementation
```
    templates\plugins\example\exampleapi.h (.cpp)
```

Implementation and slots for JavaScript
```
    templates\plugins\example\example.h (.cpp)
```

Library file to enable pre-built library inclusion in Hybrid projects (optional)
```
    templates\plugins\example\lib\4.6.3\urel\exampleapi.dll
```

JavaScript wrapper for interface slots to enable easier access and code completion in JS aware IDEs (optional)
```
    templates\plugins\example\example.js
```

7.2 Loading the plug-in

Loading the plug-in from JavaScript is simple:

```
var pluginObject = hybrid.getPlugin("example");
```

Through the pluginObject you can access all the slots exposed by the plug-in instance.

If you are planning to release your plug-in to other developers, you must create a JavaScript wrapper for your plug-in that is then included into the generated application. The wrapper should encapsulate plug-in loading and functions in a neat, trivial JavaScript file. This ensures that the API can be used by those who may not have experience of the native development. The wrapper file also serves as documentation and is compliant with the auto-completion feature in various IDEs. Developers can then use your plug-ins like any other JavaScript library (using script-tag).

7.3 Distributing the plug-in

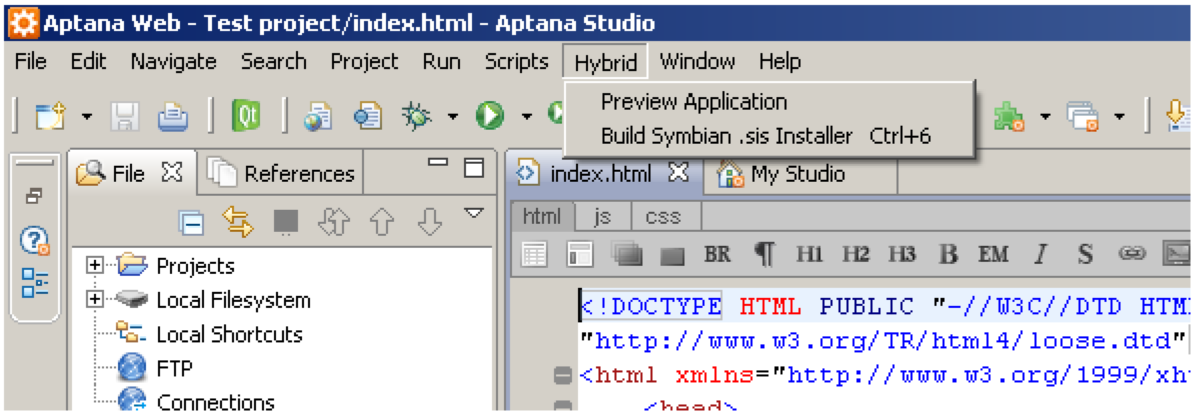
Plug-ins are automatically built inside the installation package (sis) if they are placed under the generator directory and the plug-in is selected in the Settings dialog. You may also distribute plug-ins as separate packages.

7.4 Testing the plug-in with preview functionality

The preview functionality uses the same framework and plug-ins as the mobile versions of the project. If you want to use your own plug-in in the preview application, compile the plug-in with Qt 4.6.2 for Windows (MinGW) using release configuration. The resulting DLL file should be copied to the Generator directory. Example and location plug-ins for the preview application are included by default (`exampleapi.dll` and `locationapi.dll`, respectively).

8 IDE integration

Hybrid Application Generator has a command-line interface that can be used to provide functionality to e.g. third party IDEs. An example of Aptana Studio 2.0 integration is available at Hybrid Application Generator pages at Forum Nokia Wiki. To enable the plugin, copy the `.jar` file to Aptana Studio plugins directory and restart Aptana. New functionality should appear: an icon to the Aptana toolbar, and a new menu for creating an application of the current project.



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTM
"http://www.w3.org/TR/html4/loose.dtd"
<html xmlns="http://www.w3.org/1999/xh
```

The Aptana integration is still experimental and can be used only with local SDKs. Remote generator support is a planned feature for upcoming releases.

9 Experimental: Loading Qt Mobility service

This process is similar to loading the Qt plug-in except that the plug-in reverse domain should be used (this has not yet been tested).

```
var lo = hybrid.getPlugin("com.nokia.ILocation");
```

10 Feedback

Please provide any feedback and feature requirements to us through Forum Nokia Discussion board at http://discussion.forum.nokia.com/forum/forumdisplay.php?f=221. The discussion board is actively monitored by the experts behind the Hybrid application framework.