

# Using SDK Plug-in Stationery for Xcode and the SDK Plug-in Wizard for Visual Studio .NET

The stationery and wizard for SDK plug-ins is a system for creating skeleton projects for SDK plug-ins, in Xcode and Visual Studio .NET. The system allows the developer to select from different plug-in types (menu, tool, object, object tool, and library), and creates a skeleton project which compiles and produces a working VectorWorks plug-in. The developer need not worry about compiler settings and the structure of the plug-in project – these details are handled by the stationery and wizard.

## Installation

Before attempting these steps, ensure that Xcode (on the Macintosh) and Visual Studio .NET (on Windows) are already installed on your computer.

### Macintosh

From the SDK installation, find the folder, VectorWorks SDK Plug-in, located in the ToolsMac folder. Drag the VectorWorks SDK Plug-in folder (/Library/Application Support/Apple/Developer Tools/Project Templates).

### Windows

From the SDK installation, find the folder, SDK Plug-in Wizard, located in the ToolsWin folder. Inside this folder, double-click the script entitled, InstallSDKWiz.js. This will install the SDK Wizard into the Visual Studio .NET hierarchy.

In order to build the .r resources created by the SDK Plug-in Wizard, it will be necessary to install the QuickTime SDK for Windows on your computer. This can be downloaded directly from Apple Computer at the following web site:

<http://developer.apple.com/quicktime/>.

The utility Rez.exe from the QuickTime SDK tools must be copied into the SDKLib\ToolsWin folder within the SDK.

If you create your resources on a Macintosh and flatten them to be used on Windows, you will not need the QuickTime SDK.

## Creating an SDK project

### Macintosh

In Xcode, select the File | New... menu item. In the New Project dialog that appears, scroll down to the “VectorWorks SDK Plug-in Stationery” and select the type of plug-in you wish to create. Click Next.

Type a name for the project in the Project Name edit box. Set the location to be the Sample folder within the SDK hierarchy. Click Finish.

### Windows

In Visual Studio .NET, select File | New | Project... . In the New Project dialog that appears, select the “Visual C++ Projects” option in the left (“Project Types:”) pane, and then select (single-click) the “VectorWorks SDK Plug-in” icon in the right (“Templates:”) pane.

Click the Browse button and navigate to the folder where you want the projects to be stored. Set the location to be the Sample folder within the SDK hierarchy. (The projects generated from the wizard will only work at this level in the SDK hierarchy; alternatively, you could create a folder at the same level as Sample, and select this to be the destination folder.)

If you would like to create the Mac resources on Windows, check the checkbox labeled, “Build Macintosh Resources From This Project”. When this box is checked, the Wizard will generate a .r file appropriate for the type of plug-in created by the project (for example, “Library.r” for a plug-in library). It will also create a .r file called MacResources.r, which contains an include statement to include the former plug-in. After the project has been generated, additional .r files may be included by the MacResources.r file by manually adding additional include statements to it. This might be useful if, for instance, some icons or other graphical resources were created on a Macintosh which were to be included in the build process. Those resources must first be either flattened using the Flattener utility included with the VectorWorks SDK, or converted to .r files using the Macintosh utility, DeRez.

In the Name edit box, enter the name of the project.  
Click OK to create the project.

If building Mac resources from the project, it will be necessary to make a one-time tweak in order for the project to work. In the project’s directory, there will be a batch file called BuildQTR.bat. In that file, the following line can be found:

```
set MACINTLIB=C:\QuickTime\IntsAndLibs\QTDevWin\Rincludes
```

The purpose of the line is to set the path to the location of the RIncludes folder, which is contained in the QuickTime SDK distribution. A suggested path is given; this path must be modified to point to the RIncludes folder on your system.

## Compiling the Project

The project is now ready for use. It will compile as is, and produce a working plug-in, although the plug-in has no functionality. The source code and resources may now be modified to add functionality specific to your application.

## Adding Files to the Project

### Macintosh

Files may be added to the project in the usual way. Resource files, both textual .r files and binary .rsrc files may be added to the project. Simply add them through the Project | Add Files... menu item, or simply drag them in from the Finder.

### Windows

Source code files may be added to the project in the usual way. Mac resources may be added to the project by adding additional include statements to the MacResources.r file. For example, if it is desired to add a file called OtherResources.r to the project, the following line would be added to MacResources.r:

```
#include "OtherResources.r"
```

Note that the OtherResources.r file is a textual resource file.

It is also possible to add flattened binary resource files (.qtr files) created on the Macintosh to the project. A slightly different syntax is used to include these files from the MacResources.r file. To add a file called Icons.qtr to the MacResources.r file, the following line would be added to MacResources.r:

```
include "Icons.qtr" ;
```

Note that there is no '#' symbol at the beginning of the line, and that there is a semicolon at the end of the line.

The two types of include statements may be freely mixed within the MacResources.r file. Each include statement must be on a separate line.

The Windows developer has the choice of whether to create all resources on the Macintosh and flatten the entire resource file on the Mac (see the Flattener utility, included with the VectorWorks SDK), or to build the resources on Windows using a combination of textual .r files and flattened .qtr files. The choice is purely arbitrary, based on what is most convenient. In some cases, .r files are especially attractive as they allow easy modification of strings or text using a text editor instead of a resource editor. This is particularly convenient for plug-in library projects, which require no graphical resources, and for which it is easy to maintain the library functions in the VLIB resources through a textual .r file. Additionally, revision control is easier with textual files than binary files.