ADOBE® ILLUSTRATOR® CS3

GETTING STARTED WITH ADOBE ILLUSTRATOR CS3 DEVELOPMENT

Adobe

Getting Started with Adobe Illustrator CS3 Development

Technical Note #10501

Contents

Getting Started with Adobe Illustrator CS3 Development

This document describes how to start developing using the Adobe® Illustrator® CS3 SDK. The target audience for this document is developers who need to develop plug-ins for Illustrator.

Using this document

To start developing plug-ins for Illustrator, follow these steps:

1. Set up your machine for development. Follow the instructions in "Setting up your development platform" on page 6.
2. Explore the documentation. See "Exploring the documentation" on page 7.
3. Explore, compile, and run the samples. See "Exploring the samples" on page 8.
4. Write a "HelloWorld" plug-in. Follow the instructions in "Creating your first plug-in" on page 19.
5. Start exploring the API using code snippets in "Exploring the API using code snippets" on page 26.
6. Write a plug-in using the supplied SDK plug-in project templates. See "Using the plug-in project templates" on page 26.

Terminology

The following terms are used in this document:

- *API* — Application programming interface.
- *Application* — Illustrator CS3, unless otherwise specified.
- *SDK* — Software development kit for the application.

Notational conventions

- SDK root folder — <*SDK*> indicates your locally installed SDK root folder. The actual root location depends on the installation and operating system.
- Application install folder — <*AI*> indicates the installed location of Illustrator CS3. The actual location depends on the installation and operating system.
- Microsoft® Visual Studio 8 install folder — <*VS*> indicates the installed location of Microsoft Visual Studio 8 (also known as Visual Studio 2005 or Visual C++ 2005). The actual location depends on the installation; the default is C:/Program Files/Microsoft Visual Studio 8/.
- Menu names — The right angle bracket, >, indicates menu hierarchies. For example, Tools > Options refers to the Options item on the Tools menu.
- Computer input or output (including source code) is shown in a monospaced font; for example:

```
Cannot find specified file
```

Setting up your development platform

The Illustrator CS3 SDK is available for download from the following Web site:

`http://www.adobe.com/devnet/illustrator/`

To set up your development platform on Windows®, follow these steps:

- Download the Illustrator CS3 SDK for Windows.
- Extract the contents of the downloaded zip archive to a location of your choice, to create the Illustrator CS3 SDK folder.
- Check that your Windows platform meets the basic requirements specified in the "Development Platforms" section in *Adobe Illustrator CS3 Porting Guide*.
- Configure Visual Studio as directed by the "Visual Studio Set-up" section in *Adobe Illustrator CS3 Porting Guide*.

To set up your development platform on a Macintosh®, follow these steps:

- Download the Illustrator CS3 SDK for Macintosh.
- Mount the downloaded disk image file (dmg) as a new volume.
- Copy the Illustrator CS3 SDK folder to a location of your choice.
- Check that your Macintosh platform meets the basic requirements specified in the "Development Platforms" section in *Adobe Illustrator CS3 Porting Guide*.
- Configure Xcode as directed by the "Xcode Set-up" section in *Adobe Illustrator CS3 Porting Guide*.

Exploring the documentation

Adobe Illustrator CS3 Programmer's Guide

This document describes the basic concepts of developing plug-ins for Illustrator CS3. It is aimed at all developers and is the recommended resource for plug-in developers after reading *Getting Started with Adobe Illustrator CS3 Development*.

This document is in the file <*SDK*>/docs/guides/programmers-guide.pdf.

API Reference

The API Reference provides reference documentation for the suites, classes, structs, functions, variables, and enums available in the API. It is a key resource for all developers.

This document is provided in two formats:

- <*SDK*>/docs/references/index.chm — This compiled HTML file allows text searches to be performed on the content. To view the contents on Windows, double-click the index.chm file icon in Windows Explorer to open the home page. To view the contents on a Macintosh, you need a CHM viewer (for options, see the *Adobe Illustrator CS3 Porting Guide*.
- <*SDK*>/docs/references/sdkdocs.tar.gz — This file contains the API Reference in HTML format. This format does not support text searches, but it provides the ability to view individual HTML files if desired. To view the contents, first decompress the archive, then open index.html in your browser.

Adobe Illustrator CS3 Porting Guide

This document describes how to update SDK plug-in code for Illustrator CS3. It is aimed at developers with pre-existing plug-ins, as its primary purpose is to help port existing plug-ins to the Illustrator CS3 API; however, it also contains information on known issues in the API and development platforms that may affect SDK plug-ins.

This document is in the file <*SDK*>/docs/guides/porting-guide.pdf.

API Advisor

This document reports the class, struct, and file differences between the API included on the CS2 SDK and CS3 SDK. It is aimed at all developers using the API, but it is most useful for developers updating plug-in code for Illustrator CS3.

This document is in the file <*SDK*>/docs/references/apiadvisor-ai12-vs-ai13.html.

Adobe Dialog Manager Programmer's Guide

This document provides an overview of the cross-platform API, for implementing dialog interfaces for Adobe applications. It is aimed at more experienced developers who are familiar with the Illustrator API.

This document is in the file <*SDK*>/docs/guides/adm-guide.pdf.

Using the Adobe Text Engine

This document describes how to configure your plug-in to use the Adobe text engine API, provided with the Illustrator CS3 SDK. It also has procedures for creating, editing, deleting, styling, and iterating text, with references to sample code and the API Reference.

This document is in the file <*SDK*>/docs/guides/using-adobe-text-engine.pdf.

Exploring the samples

The samples provided show how to implement key Illustrator plug-in features like tools, filters, menus, and file formats. To explore the samples, follow these steps:

1. Read "Sample features" on page 8 to get an overview of the features implemented by each sample.
2. Read "Samples guide" on page 10 to get a detailed description of each sample.
3. Read "Building and tuning the samples" on page 19 for instructions on how to compile and try out the samples.

Sample features

Figure 1 lists the features implemented by each sample. A bullet in a cell indicates the feature is implemented in the plug-in. Use this table to identify the sample most suitable as a starting point for your own plug-in: the sample that implements the most features your plug-in requires will give you the best base from which to work.

FIGURE 1 ***Sample Features***

Plug-ins / Features	ADMNonModalDialog	LiveDropShadow	MarkedObjects	MenuPlay	MultiArrowTool	Shell	SnippetRunner	TextFileFormat	TransformButtons	Tutorial	TwirlFilter	Webter
Action							•			•		
File Format								•				
Filter										•	•	
Effect											•	
Menu Item	•	•	•	•	•	•	•	•	•	•	•	•
Notifier	•		•						•			•
Panel	•		•				•		•			•
Plug-in Group		•										
Timer												
Tool-Palette Tool			•		•	•				•		
Transform									•			

For a more detailed description of the above plug-in features and what they do, see the "Types of Plug-ins" section in *Adobe Illustrator CS3 Programmer's Guide*.

Sample frameworks

In the SDK, most of the samles are based on one of two frameworks:

- The *Shell* framework is provided by the Shell sample and provides a basic framework for building a plug-in that contains one or more of the plug-in features listed in Figure 1. To use this framework, copy the Shell project and tailor it to your needs.
- The *Plugin/Suites* framework is based on a more class-based design, as both Plugin and Suites are classes at the core of this framework. The Plugin class constructs the basic requirements for an Illustrator plug-in, using the Suites class to acquire and release the required suites. To use this framework, create a new project, and add Plugin.cpp and Suites.cpp to your new project from <*SDK*>/samplecode/common/source/. Then create a class in your project that is a subclass of the Plugin class; this class should implement the required functionality of the plug-in.

Samples guide

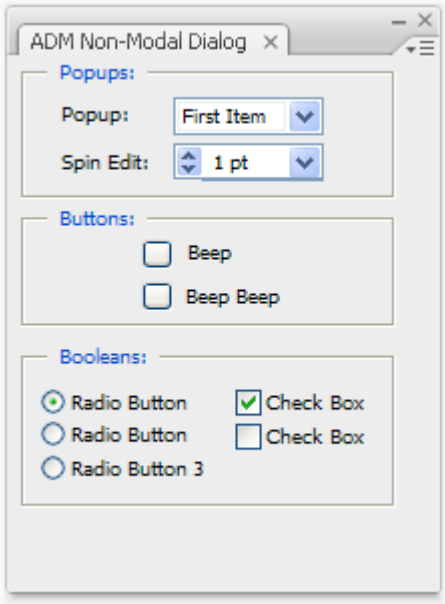
ADMNonModalDialog

This plug-in demonstrates how to create a panel using ADM. It can be used as a starter project for any plug-in that implements a panel. It is based on the Shell framework.

Menu: Window > SDK > ADM Non Modal Dialog

Panel: See Figure 2.

FIGURE 2 ***ADM non-modal dialog sample panel***



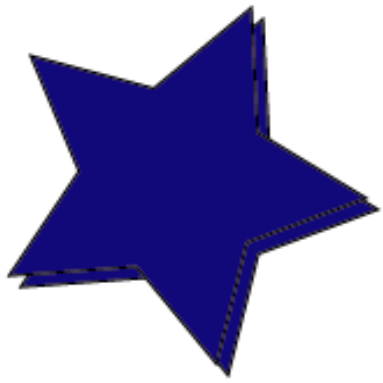
LiveDropShadow

This plug-in uses the Plugin group suite. It creates “live” drop shadows on objects. It works like the current DropShadow filter, except the shadow follows any edits to the object. It is based on the Shell framework.

Menu: Object > SDK > Live Drop Shadow

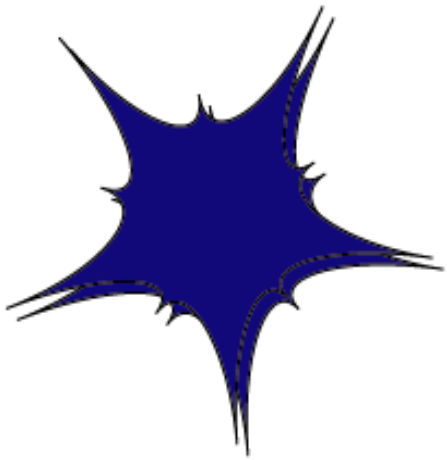
Figure 3 shows an object with a live drop shadow.

FIGURE 3 ***Object with live drop shadow applied***



To demonstrate the "live" nature of the drop shadow, Figure 4 shows the result of changing the shape of the object using the Filter > Distort > Pucker & Bloat... tool.

FIGURE 4 ***Updated drop shadow after changing the object's shape***



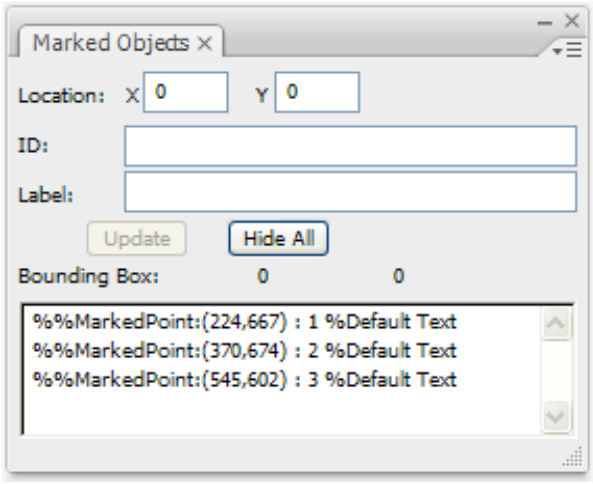
MarkedObjects

This plug-in implements a way to mark diagrams with art, to help lay out books during publishing. It creates the art via a tool in the Anchor group in the tool palette; when the tool is selected, you click on the document to add a new marked object. You also can view and edit the details of these marked objects from the Marked Objects dialog.This plug-in is based on the Plugin/Suites framework.

Menu: Window > SDK > Marked Objects

Panel: See Figure 5.

FIGURE 5 ***Marked objects sample panel***



The sample adds the tool shown in Figure 6, in the Pen Tool group on the Tools palette.

FIGURE 6 ***Icon representing the tool added to the tools palette by the marked object sample***

Figure 7 is an example of an object created by the Marked Object tool. The object is a group containing two art items representing the cross and two text items representing the label and ID.

FIGURE 7 ***Sample object created by the marked object tool***

MenuPlay

This plug-in demonstrates how to add and manipulate menu items. It is based on the Shell framework.

Menu: Window > SDK > Menu Play

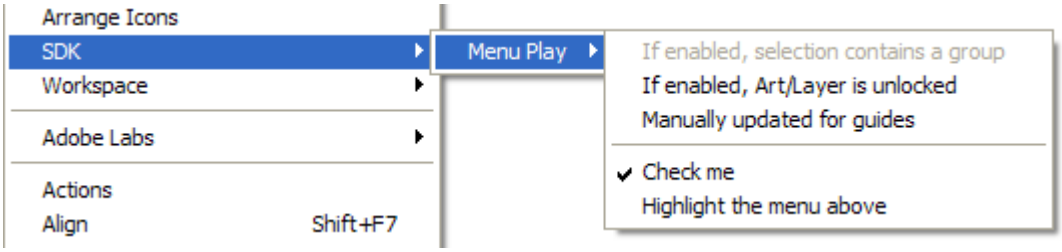
The types of menu manipulation demonstrated by this sample include the following:

- Highlighting menu items when particular item types are selected.
- Highlighting menu items when the art layer has a particular state.
- Changing the text displayed by the menu item when an item has a particular state.

- Allowing a menu item to affect the availability of another menu item.
- Allowing a menu item to display an icon.

Figure 8 shows sample menu items for this plug-in.

FIGURE 8 ***Menu play sample menu items***

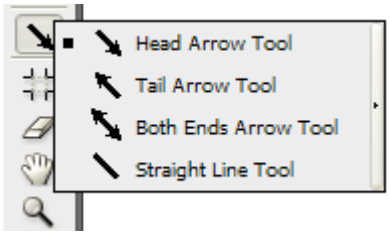


MultiArrowTool

This plug-in demonstrates how to add multiple tools to the tool palette, change icons, and handle dynamic redraw of tool modifications. It is based on the Shell framework.

This sample creates its own tool group on the tools palette, with the most recently selected arrow icon at the front of the group. See Figure 9.

FIGURE 9 ***Tools added by the multi-arrow tool sample***



Once one of the arrow tools is selected, the user can click and drag in the document to create a new path in the shape of an arrow. The Head Arrow tool creates an arrow with the head at the end of the path; the Tail arrow tool, at the beginning of the path.

Shell

This plug-in provides a basic framework for many plug-in types. The Shell plug-in adds a tool to the tool palette and a menu item to the Window menu that, when selected, display a basic dialog. To create a new plug-in, simply duplicate this project and add your plug-in-specific code. The Shell framework supports plug-ins that create and work with the following:

- Actions
- File formats
- Filters

- Menu items
- Notifiers
- Panels
- Plug-in groups
- Timers
- Tool-palette tools
- Transforms

Menu: Window > SDK > Shell

The icon in Figure 10 is located within the Hand Tool group on the tools palette. This tool has no function other than demonstrating how to display an icon on the tools-palette.

FIGURE 10 Icon representing the tool added to the tools palette by the shell sample

The Shell sample also displays an alert when its menu item is selected.

SnippetRunner

A code snippet in Illustrator is a source file, containing a class, in which you can quickly and easily prototype SDK API calls.

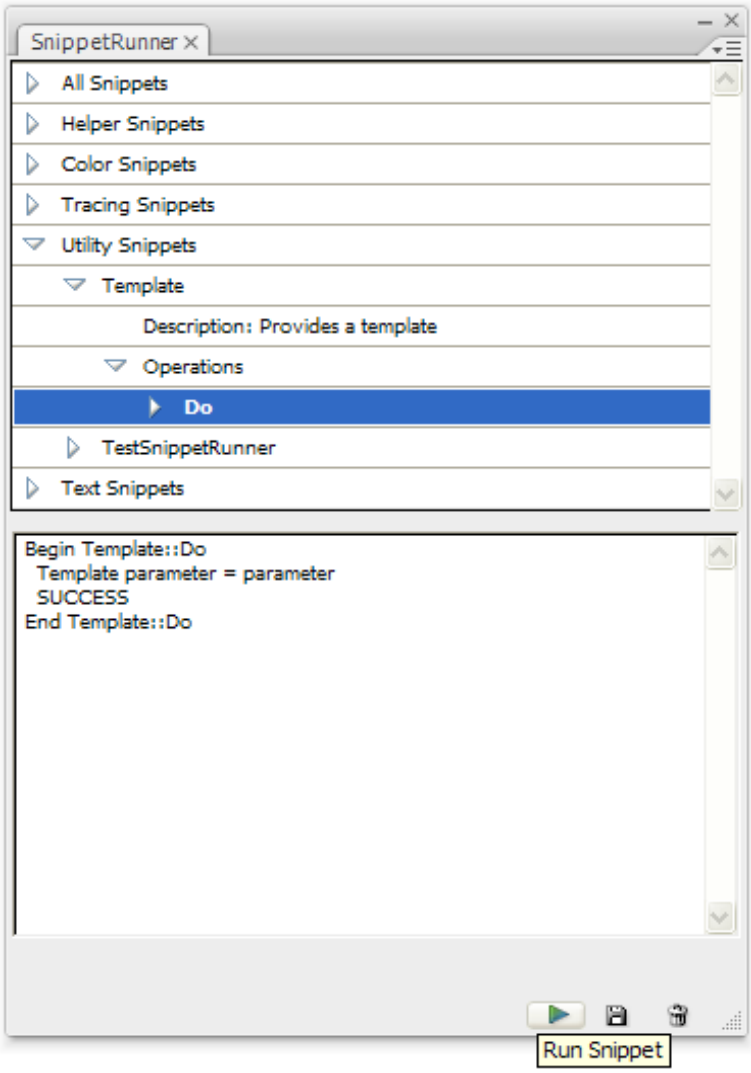
The SnippetRunner plug-in hosts and runs code snippets within Illustrator CS3. The framework it provides includes the following:

- A user interface that allows code snippets to be chosen and run.
- A parameter-input mechanism.
- A log-output mechanism.
- Unit testing.

Menu: Window > SDK > SnippetRunner

Figure 11 shows the SnippetRunner user interface.

FIGURE 11 ***The SnippetRunner user interface***



The user-interface controls are as follows:

- Run button — Runs the selected snippet or snippet operation.
- Disk button — Saves the log.
- Trash button — Clears the log.
- Panel flyout menu — Allows the user to specify the parameter input mode, toggle unit testing, and set the path to the assets folder. Three parameter-input modes are available: Default Parameters (parameters are provided in the code), Prompt Parameters (prompts are provided for parameter values), and CSV Parameters (values are provided in a CSV string).

NOTE: The code snippets can be run or unit tested from the operation, snippet, or category level.

The assets folder contains the files used for unit testing; it also is the location where modified unit-test documents are saved. This folder is set up the first time a code snippet is run, and it remains the same until either the preference is deleted or the assets folder is changed through the Set Assets Folder option in the panel flyout menu.

The location of the assets folder required to run SnippetRunner unit tests is <*SDK*>/samplecode/CodeSnippets/examplefiles. This folder contains a .ai file and supporting image files designed to provide the required context for several snippet operations.

See the <*SDK*>/samplecode/CodeSnippets/ folder for the list of snippets provided with SnippetRunner, and see the brief snippet descriptions in the SnippetRunner user interface for details of what each snippet does.

NOTE: An important code snippet to note is SnpTemplate which provides a basic framework for new snippets and instructions on how to tailor the template to the new snippets needs.

TextFileFormat

This plug-in shows how to add new file formats to Illustrator. It adds the "All Text as TEXT" format to Illustrator's Open dialog and Save dialog. It also adds the "Selected Text as TEXT" format to Illustrator's Save dialog.

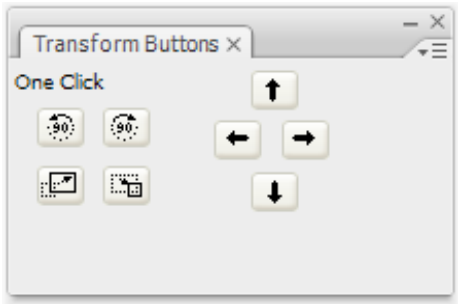
TransformButtons

This plug-in uses the TransformArt suite (see AITransformArtSuite). It creates a palette that executes one-click transformations to art objects. It is based on the Shell framework.

Menu: Window > SDK > Transform Buttons

Panel: See Figure 12.

FIGURE 12 Transform Buttons sample panel



Tutorial

This plug-in illustrates the basics of plug-in development. It is the sample used in the "Tutorial" section of *Adobe Illustrator CS3 Programmers Guide*.

Menu: Filter > SDK > Tutorial...

This sample creates an art item from the data passed in from the dialog shown in Figure 13, which displays its default values. The sample also defines minimum and maximum values for each data item.

FIGURE 13 ***Modal dialog used by the tutorial sample to collect parameters from the user***

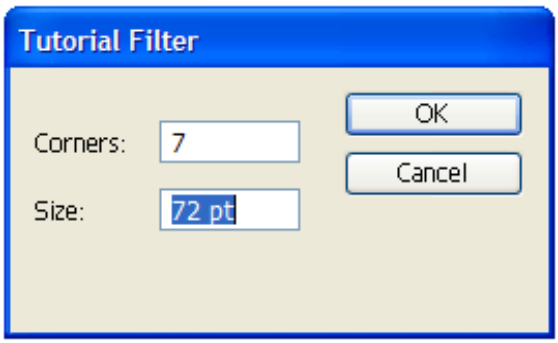


Figure 14 shows art created by the default values.

FIGURE 14 ***Example of art created by the Tutorial sample's filter***

This sample also adds the tool shown in Figure 15.

FIGURE 15 ***Tutorial line tool icon***

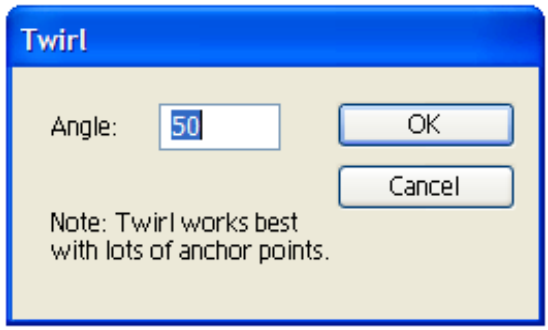
TwirlFilter

This plug-in demonstrates how to create filters and live effects that add menu items and manipulate selected artwork. It is based on the Shell framework.

Menu: Filter > SDK > Twirl... & Effect > SDK > Twirl...

This sample adds menu items to the Filter and Effect menus, essentially implementing the functionality of a filter and an effect. The dialog is the same for both instances and performs the same change on the selected item. The selected item is rotated by a radian value calculated from the degree value passed in through the dialog. Figure 16 shows the default degree value.

FIGURE 16 ***Modal dialog used by the Twirl filter sample to collect data from the user***



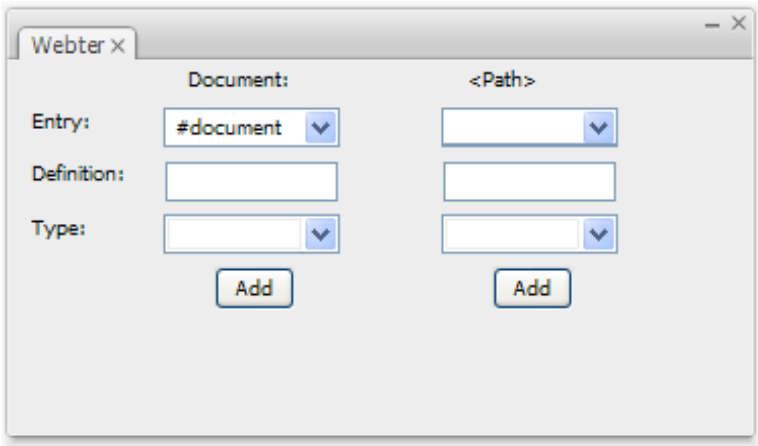
Webter

This plug-in manipulates the dictionary for document and art objects (see AIDictionarySuite). It is based on the Plugin/Suites framework.

Menu: Window > SDK > Webter

Panel: See Figure 17.

FIGURE 17 ***Webter sample panel***



Building and running the samples

To build all the samples, use the projects provided in the <*SDK*>/samplecode/MasterProjects/ folder. After compiling and linking, the plug-in binaries can be found in the <*SDK*>/samplecode/Output/ folder. For instructions on getting Illustrator to load these samples, see the "Getting Illustrator to Load your Plug-in" section of *Adobe Illustrator CS3 Porting Guide*.

To debug a sample follow the instructions in the "Running and Debugging" section of *Adobe Illustrator CS3 Porting Guide*.

Creating your first plug-in

This section describes how to write from scratch a plug-in that extends Illustrator in a very basic way. The instructions create a HelloWorld plug-in for Illustrator, which pops an alert when Illustrator starts up or shuts down.

Creating a HelloWorld plug-in for Illustrator on Windows

Create a new project

1. Start Visual C++.
2. Choose File > New > Project...
3. In the New Project dialog, do the following:

- Expand the Visual C++ filter.
- In the Project Types window, highlight Win32.

- In the Templates window, select Win32 Project.
- Set the Name property of the project to HelloWorld.
- Set the Location property of the project to <*SDK*>/samplecode.
- Deselect Create Directory For Solution.
- Click OK.

4. In the Win 32 Application Wizard, do the following:

- Select Application Settings.
- Under Application Type, select DLL.
- Under Additional Options, select Empty Project.
- Click Finish.

Add a source file

1. In the Solution Explorer, right-click the Source Files folder.
2. Choose Add > New Item...
3. In the Add New Item dialog, do the following:

- Expand the Visual C++ filter.
- Highlight Code, and select C++ File (.cpp).
- Set the Name property of the file to HelloWorld.cpp.
- Click Add.

4. Copy the code from Figure 18 to the source file, and save.

FIGURE 18 HelloWorld.cpp

```
#include "IllustratorSDK.h"

// Tell Xcode to export the following symbols
#if defined(__GNUC__)
#pragma GCC visibility push(default)
#endif

// Plug-in entry point
extern "C" ASAPI ASErr PluginMain(char* caller, char* selector, void* message);

// Tell Xcode to return to default visibility for symbols
#if defined(__GNUC__)
#pragma GCC visibility pop
#endif

extern "C" ASAPI ASErr PluginMain(char* caller, char* selector, void* message)
{
    ASErr error = kNoErr;
    SPBasicSuite* sSPBasic = ((SPMessageData*)message)->basic;

    if(sSPBasic->IsEqual(caller,kSPInterfaceCaller))
    {
        ADMBasicSuite10 *sADMBasic = nil;
        error = sSPBasic->AcquireSuite(kADMBasicSuite,
                kADMBasicSuiteVersion10,(const  void**)&sADMBasic);
        if(sSPBasic->IsEqual(selector,kSPInterfaceStartupSelector))
            sADMBasic->MessageAlert("Hello World!");

        else if(sSPBasic->IsEqual(selector,kSPInterfaceShutdownSelector))
            sADMBasic->MessageAlert("Goodbye World!");
        error = sSPBasic->ReleaseSuite(kADMBasicSuite,kADMBasicSuiteVersion10);
    }

    return error;
}
```

Add a resource file

1. In the Solution Explorer, right-click the Resource Files folder.
2. Choose Add > New Item...
3. In the Add New Item dialog, do the following:

- Expand the Visual C++ filter.
- Highlight Resource, and select Resource File (.rc).
- Set the Name property of the file to HelloWorld.rc.
- Click Add.

4. Close the Resource View window.

5. In the Solution Explorer, right-click HelloWorld.rc.
6. Select View Code.
7. Just before the line containing #ifdef APSTUDIO_INVOKED, insert the code from Figure 19.
8. Save the file.

FIGURE 19 HelloWorld.rc

```
#define kMySDKPluginName "HelloWorld"
#define PIPL_PLUGIN_NAME kMySDKPluginName

//////////////////////////////////////////////////////////////////////////////
//
// PIPL
//

#define LC(a,b,c,d)     #d, #c, #b, #a

16000  PiPL  DISCARDABLE
BEGIN
    0x0001,                     /* Reserved (for Photoshop) */
    0L,                         /* kCurrentPiPLVersion */
    4L,                         /* Property Count */

    LC(A,D,B,E),                /* vendorID */
    LC(k,i,n,d),                /* propertyKey = PIKindProperty */
    0L,                         /* propertyID */
    4L,                         /* propertyLength */
    LC(S,P,E,A),                /* propertyData  */

    LC(A,D,B,E),                /* vendorID */
    LC(i,v,r,s),                /* propertyKey = PISPVersionProperty */
    0L,                         /* propertyID */
    4L,                         /* propertyLength */
    2L,                         /* propertyData */

    LC(A,D,B,E),                /* vendorID */
    LC(w,x,8,6),                /* propertyKey = PIWin32X86CodeProperty */
    0L,                         /* propertyID */
    12L,                        /* propertyLength */
    "PluginMain\0\0",           /* propertyData = Entry Point  */
                                /* (Long Word padded C String) */

    LC(A,D,B,E),                /* Vendor Key */
    LC(p,i,n,m),                /* propertyKey = PIPluginNameProperty  */
    0L,                         /* propertyID */
    12L,                        /* propertyLength */
    PIPL_PLUGIN_NAME "\0\0"     /* propertyData = Plug-in name (padded to 4
    bytes) */
END
```

Configure the project settings

1. Right-click the project name in the Solution Explorer, and select Properties to open the Property Pages dialog.
2. In the project Property Pages dialog, expand the Configuration Properties filter.
3. Expand the C/C++ filter.
4. Choose Configuration Properties > C/C++ > General.
5. In the Additional Include Directories field, enter the following:

```
..\..\illustratorapi\adm,..\..\illustratorapi\illustrator,..\..\illustratorapi\
pica_sp
```

6. Choose Configuration Properties > C/C++ > Code Generation.
7. Set Struct Member Alignment to 4 Bytes (/Zp4).
8. Expand the Linker filter.
9. Choose Configuration Properties > Linker > General.
10. Set Output File to ..\output\win\debug\HelloWorld.aip
11. Choose Configuration Properties > Linker > Input.
12. Set Module Definition File to ..\common\win\PluginMain.def
13. Click Apply, then OK.

Build the project

1. Only a debug configuration is set up in the Visual C++ projects; however, ensure Debug is selected in the drop-down list.
2. Build the project by choosing Build > Build HelloWorld, and fix any build errors.

Creating a HelloWorld plug-in for Illustrator on the Macintosh

Create a new project

1. Start Xcode.
2. Choose File > New Project...
3. In the New Project Assistant, select Empty Project.
4. Click Next.
5. Set the Project Name to HelloWorld.
6. Set the Project Directory to <*SDK*>/samplecode.

7. Click Finish.
8. Under Groups & Files, right-click the project name.
9. Choose Add > New Group.
10. Name the new group Source.
11. Add another new group, called Resources.

Add a target

1. Under Groups & Files, right-click Targets.
2. Choose Add > New Target...
3. In the New Target Assistant, select Carbon > Loadable Bundle.
4. Click Next.
5. Set the Target Name to HelloWorld.
6. Click Finish.
7. Select the Build tab of the Target Info dialog that opened when the target was created.
8. Scroll down to the Wrapper Extension setting, and set it to aip.
9. Close the Target Info dialog.
10. Right-click the new target.
11. Choose Add > New Build Phase > New Build ResourceManager Resources Build Phase.

Add a source file

1. Under Groups & Files, right-click the Source folder.
2. Choose Add > New File...
3. Select Empty File in Project.
4. Click Next.
5. Set the File Name to HelloWorld.cpp.
6. Check off the Target.
7. Click Finish.
8. Copy the code from Figure 18 to the source file, and save.

Add a resource file

1. Under Groups & Files, right-click the Resources folder.
2. Choose Add > New File...
3. Select Empty File In Project.
4. Click Next.
5. Set the File Name to HelloWorld.r.
6. Check off the Target.
7. Click Finish.
8. Copy the following to the new resource file:

```
#define PIPL_PLUGIN_NAME "HelloWorld"
#include "Plugin.r"
```

Configure the project settings

1. Under Groups & Files, right-click the project icon.
2. Select Get Info.
3. Set SDK Path to /Developer/SDKs/MacOSX10.4u.sdk.
4. Set Build Products Path to ./../output/mac/$(AI_CONFIGURATION).
5. Set Header Search Paths to ../common/** ../../illustratorapi/**.
6. Check off Force Package Info Generation.
7. Set Info.plist File to ../common/mac/Info.plist.
8. Check off Preprocess Info.plist File.
9. Set Info.plist Preprocessor Prefix File to ../common/includes/SDKDef.h.
10. Close the Project Info dialog.

Build the project

1. Select the debug configuration by choosing Debug from the Active Build Configuration drop-down list.
2. Build the project by choosing Build > Build, and fix any build errors.

Loading an Illustrator plug-in

After compiling and linking, the plug-in binary file is in the following locations:

- Windows: <*SDK*>/samplecode/Output/win/debug/
- Macintosh: <*SDK*>/samplecode/Output/Mac/debug/

1. To ensure the plug-in binary is in a location searched by Illustrator for plug-ins to load, follow the instructions in the "Getting Illustrator to Load your Plug-in" section of *Adobe Illustrator CS3 Porting Guide*.
2. Start Illustrator CS3. During start-up, an alert is popped, displaying a short greeting. Once the application has started, your plug-in's name should be listed in the Help > System Info... dialog. During shut-down, an alert is popped, displaying another short message.
3. For instructions on how to debug your plug-in, see the "Running and Debugging" section in *Adobe Illustrator CS3 Porting Guide*.

Exploring the API using code snippets

Using code snippets to explore the API is quick and easy with the provided SnippetRunner plug-in, see "SnippetRunner" on page 14. All you have to do is add a single file to both the Xcode and Visual C++ project files, and a template code snippet is provided - <*SDK*>/samplecode/codesnippets/SnpTemplate.cpp which contains the required hooks to get the new code snippet running in SnippetRunner.

With the significantly reduced time required to create a code snippet rather than an entire plug-in, more time can be spent exploring the API.

Follow the instructions provided in <*SDK*>/samplecode/codesnippets/SnpTemplate.cpp to create and tailor a new code snippet to your requirements.

NOTE: You will have to acquire any suites your code snippet requires that are not already provided with SnippetRunner. There are instructions in SnpTemplate.cpp on where these should be added.

Using the plug-in project templates

The templates discussed in the following sections are in the <*SDK*>/samplecode/Templates/ folder. To create a starting point for your plug-in, we recommend you use the templates and the instructions below, instead of the instructions in "Creating your first plug-in" on page 19, as the templates give you a new project with the same configurations as the sample plug-ins.

Visual C++ template

1. Quit Visual Studio if it is running.
2. Copy the <*SDK*>/samplecode/Templates/Win/IllustratorSDKPlugin/ folder to <*VS*>/VC/VCWizards/AppWiz/.
3. Copy the <*SDK*>/samplecode/Templates/Win/IllustratorPlugin/ folder and IllustratorPlugin.vsz file to <*VS*>]/VC/vcprojects/.
4. Start Visual Studio.
5. Choose File > New > Project...
6. In the Project Types pane of the New Project dialog, select Visual C++ > IllustratorPlugin.
7. In the Templates pane, select SDKProject.
8. Set the Name of the project to a name of your choice.
9. Set the Location of the project to the <*SDK*>/samplecode folder. (If you save to a different location, you will have to update the relative paths.)
10. Ensure Create Directory For Solution is not checked. (If this option is selected, the relative paths will have to be updated, as the project file will not be at the top level.)
11. Select OK.
12. In the Solution Explorer, right-click IllustratorSDK.cpp in the Source Files > Shared folder, and select Properties.
13. Select Configuration Properties > C/C++ > Precompiled Headers, and set Create/Use Precompiled Header to Create Precompiled Header (/Yc).
14. Click Apply, then OK.
15. Build the project and load it into Illustrator, following the instructions in "Loading an Illustrator plug-in" on page 26.

NOTE: There is no release configuration created by the template.

Xcode template

1. Quit Xcode if it is running.
2. Copy the <*SDK*>/samplecode/Templates/Mac/IllustratorPlugin folder to <*startup disk*/Library/Application Support/Apple/Developer Tools/Project Templates/.]
3. Start Xcode.
4. Choose File > New Project...
5. In the New Project dialog, select IllustratorPlugin > SDKProject, and click Next.

6. Give the project a name of your choice.
7. Set the Project Directory to <*SDK*>/samplecode, then click Finish.
8. Build the project and load it into Illustrator, following the instructions in "Loading an Illustrator plug-in" on page 26.