

ADOBE® ILLUSTRATOR® CS4



ADOBE ILLUSTRATOR CS4 PORTING GUIDE



© 2008 Adobe Systems Incorporated. All rights reserved.

Adobe Illustrator CS4 Porting Guide

Technical Note #10500

If this guide is distributed with software that includes an end user agreement, this guide, as well as the software described in it, is furnished under license and may be used or copied only in accordance with the terms of such license. Except as permitted by any such license, no part of this guide may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, recording, or otherwise, without the prior written permission of Adobe Systems Incorporated. Please note that the content in this guide is protected under copyright law even if it is not distributed with software that includes an end user license agreement.

The content of this guide is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Adobe Systems Incorporated. Adobe Systems Incorporated assumes no responsibility or liability for any errors or inaccuracies that may appear in the informational content contained in this guide.

Please remember that existing artwork or images that you may want to include in your project may be protected under copyright law. The unauthorized incorporation of such material into your new work could be a violation of the rights of the copyright owner. Please be sure to obtain any permission required from the copyright owner.

Any references to company names in sample templates are for demonstration purposes only and are not intended to refer to any actual organization.

Adobe, the Adobe logo, and Illustrator are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries. Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Apple, Mac OS, and Macintosh are trademarks of Apple Computer, Incorporated, registered in the United States and other countries. All other trademarks are the property of their respective owners.

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA. Notice to U.S. Government End Users. The Software and Documentation are "Commercial Items," as that term is defined at 48 C.F.R. §2.101, consisting of "Commercial Computer Software" and "Commercial Computer Software Documentation," as such terms are used in 48 C.F.R. §12.212 or 48 C.F.R. §227.7202, as applicable. Consistent with 48 C.F.R. §12.212 or 48 C.F.R. §§227.7202-1 through 227.7202-4, as applicable, the Commercial Computer Software and Commercial Computer Software Documentation are being licensed to U.S. Government end users (a) only as Commercial Items and (b) with only those rights as are granted to all other end users pursuant to the terms and conditions herein. Unpublished-rights reserved under the copyright laws of the United States. Adobe Systems Incorporated, 345 Park Avenue, San Jose, CA 95110-2704, USA. For U.S. Government End Users, Adobe agrees to comply with all applicable equal opportunity laws including, if appropriate, the provisions of Executive Order 11246, as amended, Section 402 of the Vietnam Era Veterans Readjustment Assistance Act of 1974 (38 USC 4212), and Section 503 of the Rehabilitation Act of 1973, as amended, and the regulations at 41 CFR Parts 60-1 through 60-60, 60-250, and 60-741. The affirmative action clause and regulations contained in the preceding sentence shall be incorporated by reference.



Contents

Introduction	5
Terminology	5
Notational conventions	5
Illustrator API Changes	6
Annotation enhancements	6
Multiple artboards	6
FXG	10
Gradients	10
Isolation-mode improvements	10
OWL 2.0	11
Smart Guide enhancements	14
ATE changes	15
Menu clean-up	16
Miscellaneous changes	16
Versioned suites	20
Legacy files removed	20
SDK Changes	21
Sample plug-in framework changes	21
Shell plug-in is now legacy	23
Changes to common folder structure	23
Removal of unused source and resource files	25
Removal of Mac OS binary resources	26
Removal of Mac OS GetCursor and SetCursor functions	28
MarkedObjects preferences dialog uses EVE resource definitions	29
Porting Case Study	30
Porting MarkedObjects on Windows	30
Porting MarkedObjects on Mac OS	30

Adobe Illustrator CS4 Porting Guide

This document describes how to update your SDK plug-in code and development environments for Adobe® Illustrator® CS4. It details changes in the public API and other aspects of the SDK since the CS3 release.

Introduction

To begin porting, follow these steps:

1. Ensure that your system meets the basic requirements specified in “Development platforms” section in *Getting Started with Adobe Illustrator CS4 Development*.
2. Install the SDK.
3. Examine the documentation, then compile and run the samples.
4. To port your own plug-in projects and code, follow the recommended procedures.

Terminology

The following terms are used in this document:

- *API* — Application programming interface.
- *Application* — Illustrator CS4, 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.
- *Menu names* — The right angle bracket, >, indicates menu hierarchies. For example, Tools > Options refers to the Options menu item on the Tools menu.
- *Computer input or output* (including source code) is shown in a monospace font; for example:

```
Cannot find specified file
```

Illustrator API Changes

This section summarizes the important changes to the API since the prior release.

Annotation enhancements

As an architecture improvement, annotation drawing in Illustrator CS4 can now be carried out by the AGM Compositor instead of the GDI and QuickDraw platform APIs.

The following changes were made to the public API.

New AIAnnotatorDrawerSuite

This is the recommended suite to draw annotations in response to an AIAnnotatorMessage. Annotations using ADMDrawer and AGMSuite are still supported in CS4, but future releases may disallow annotation drawing using platform APIs like GDI/GDI+ on Windows and QuickDraw and Quartz on Mac OS.

New kDrawOnInactiveDocuments flag

As part of the annotation enhancements, plug-ins can now opt-in to receive annotation notifications on inactive documents. Previously, notifications were sent to all documents; now, by default, notifications are sent only to the active document. If the kDrawOnInactiveDocuments flag is set, notifications are sent to all documents.

AIAnnotatorSuite API changes

The following API signatures were changed to accept an AIAnnotatorOptionsFlags type. This type contains flags that determine how an annotator is drawn.

- AIAnnotatorSuite::GetAnnotatorOptions
- AIAnnotatorSuite::SetAnnotatorOptions

The signatures have changed from:

```
AIAPI AIErr (*GetAnnotatorOptions) (AIAnnotatorHandle notifier, long *flags);  
AIAPI AIErr (*SetAnnotatorOptions) (AIAnnotatorHandle notifier, const long flags);
```

to:

```
AIAPI AIErr (*GetAnnotatorOptions) (AIAnnotatorHandle notifier,  
AIAnnotatorOptionsFlags *flags);  
AIAPI AIErr (*SetAnnotatorOptions) (AIAnnotatorHandle notifier, const  
AIAnnotatorOptionsFlags flags);
```

Multiple artboards

The new, multiple-artboards feature allows the user to create and edit artwork across several artboards, then export each artboard as a JPEG, PNG, or SWF file. The maximum number of artboards allowed is 100. During the creation of a new document, the user can specify the

number of artboards the document should have and their layout. If there is only one artboard in the document, the behavior is similar to CS3.

Multiple artboards are a more advanced implementation of the crop-area feature added in CS3; therefore, the APIs are accessed through AICropAreaSuite and the new AICropAreaRangeSuite.

The following changes were made to the public API.

New suite: AICropAreaRangeSuite

This suite was added as part of the multiple-artboard feature in Illustrator CS4. AICropAreaSuite is used to create artboards. AICropAreaRangeSuite facilitates the extraction of a range of artboards using an AICropAreaRangeHandle.

Properties removed from AICropArea struct

In CS4, the properties that determine whether to show crop-area rulers and screen edges were removed from the AICropArea struct.

The show crop-area rulers property was removed because artboard rulers are no longer tied to artboard properties. Artboard rulers are visible only for the active artboard, if this option is enabled through the new APIs in the AIDocumentViewSuite. See [“APIs added to AIDocumentViewSuite” on page 7](#).

The show screen-edge property was removed in CS4 because is no longer being used.

AICropAreaSuite new notifier source

The source enum in AICropArea.h has a new option, kUndoRedo. The kAIDocumentCropAreaModifiedNotifier notifier is now sent on undo/redo modifications.

APIs added to AIDocumentViewSuite

The following APIs were added to AIDocumentViewSuite:

- AIDocumentViewSuite::IsArtboardRulerVisible
- AIDocumentViewSuite::SetArtboardRulerVisible

For details, see AIDocumentViewSuite in *Adobe Illustrator CS4 API Reference*.

APIs removed from AIDocumentListSuite

Two APIs were removed from the AIDocumentListSuite:

- AIDocumentListSuite::ActivateStartupDocument
- AIDocumentListSuite::GetStartupDocument

These APIs used the AIShutdownDocumentFlag to define which startup template to use for the new document. Now, all relevant information is passed to AIDocumentListSuite::New in the AINewDocumentPreset struct.

AINewDocumentPreset struct changes

As part of the implementation of the multiple artboard feature, the following attributes were added to the AINewDocumentPreset struct in AIDocumentList.h. These are used to set additional controls in the New Document dialog.

- AIArtboardLayout docArtboardLayout
- AReal docArtboardSpacing
- ASInt32 docArtboardRowsOrCols
- ASInt32 docNumArtboards

For more information about these new attributes, see AIDocumentList.h.

New enum: AIArtboardLayout

A new AIArtboardLayout enum was added to AIDocumentList.h. It defines how the new document will display the number of artboards you have chosen. The following types are defined in this enum:

- kAIArtboardLayoutGridByCol
- kAIArtboardLayoutGridByRow
- kAIArtboardLayoutCol
- kAIArtboardLayoutRow
- kAIArtboardLayoutRLGridByCol
- kAIArtboardLayoutRLGridByRow
- kAIArtboardLayoutRLRow

For more information about the AIArtboardLayout types, see AIDocumentList.h.

AIStartupDocumentFlag type removed

The AIStartupDocumentFlag enum was removed. Previously, the start-up documents specified by this flag were used as the templates for creating new documents using AIDocumentListSuite::GetStartupDocument and AIDocumentListSuite::ActivateStartupDocument APIs, which were removed. See [“APIs removed from AIDocumentListSuite” on page 7](#).

Instead, use AIDocumentListSuite::New API to create a new document. The AINewDocumentPreset struct passed to this API contains all the information required about the new document. See [“AINewDocumentPreset struct changes” on page 8](#).

FlashExportOption added

A new FlashExportOption, exportMultipleArtboards, facilitates the export of each artboard in the document to a SWF file. For details, see AIFlashPrefs.h.

Enums added to AIFileFormatSuite

Two new enums were added to AIFileFormatSuite to support multiple artboards:

- **AIFFExtendedOptions** — This specifies whether the file format supports the save/export of multiple artboards. If it is supported, the extended options in the save/export dialog are active. The options available are **kNoExtendedOptions** and **kSaveMultiArtboards**.
- **eFFOperationOptions** — This is a flag passed in the AIFileFormatMessage which specifies whether the multiple artboards should be saved separately. For details, see AIFileFormat.h.

AIFileFormatSuite API changes

The following API signatures were changed to support the extended options, which specify whether a file format saves/exports multiple artboards:

- AIFileFormatSuite::AddFileFormat
- AIFileFormatSuite::AddFileFormatEx

The signatures were changed from:

```
AI_API AI_Error (*AddFileFormat) (SPPluginRef self, const char *name,
PlatformAddFileFormatData *data, long options, AIFileFormatHandle *fileFormat);
AI_API AI_Error (*AddFileFormatEx) (SPPluginRef self, const char *name,
PlatformAddFileFormatExData *dataEx, long options, AIFileFormatHandle *fileFormat);
```

to:

```
AI_API AI_Error (*AddFileFormat) (SPPluginRef self, const char *name,
PlatformAddFileFormatData *data, long options, AIFileFormatHandle *fileFormat, long
extendedOptions);
AI_API AI_Error (*AddFileFormatEx) (SPPluginRef self, const char *name,
PlatformAddFileFormatExData *dataEx, long options, AIFileFormatHandle *fileFormat,
long extendedOptions);
```

The new required parameter, **long extendedOptions**, is a logical OR of **AIFFExtendedOptions**. (For more information on **AIFFExtendedOptions**, see [“Enums added to AIFileFormatSuite” on page 9](#)).

APIs added to AIFileFormatSuite

Two APIs were added to the AIFileFormatSuite to get and set the extended options flag of a file format:

- AIFileFormatSuite::GetFileFormatExtendedOptions
- AIFileFormatSuite::SetFileFormatExtendedOptions

The extended options flag is a logical OR of **AIFFExtendedOptions**. For details, see [“Enums added to AIFileFormatSuite” on page 9](#) and the AIFileFormatSuite in *Adobe Illustrator CS4 API Reference*.

FXG

A new file format, FXG, was added to Illustrator CS4. FXG is an XML dialect used to describe the visual components of a graphic. It can be read and written by Illustrator, as well as authored by hand by a designer or programmer. The goal of this new format is to improve designer/developer workflow by providing round-tripping capabilities between Illustrator and other XML-based applications.

The AIFXGFileFormatSuite provides extension points for exporting Illustrator documents to FXG files and output streams. For details, see AIFXGFileFormat.h in the API.

Gradients

New features added to the existing radial gradient, including an aspect-ratio parameter and an angle (previously available only with linear gradients), give the effect of an elliptical gradient.

AIArtStylePaintData struct changes

A new AReal value, which specifies the actual aspect ratio of a gradient, was added to the AIArtStylePaintDataStruct. This ratio does not include any object transformations. This new value was added to support the new elliptical-gradient feature in Illustrator CS4.

NOTE: The aspect ratio of a two-dimensional object is the ratio between the vertical and horizontal spreads of the ellipse. For example, if the elliptical gradient measures 50 units in the X axis and 100 units in the Y axis, the aspect ratio is 2 (100/50).

The matrix field in the AIGradientStyle struct includes any gradient aspect ratio applied, along with other transformations applied on the gradient. The aspect-ratio information is prepended to the matrix in AIGradientStyle that represents any transformations to the gradient resulting from transforming the object.

AIGradientStop struct changes

Gradient stops have a new attribute for opacity. A new AReal value, which specifies the opacity for a gradient stop, was added to the AIGradientStop struct. The alpha values blend from one stop to another in the gradient.

API added to AIGradientSuite

A new API was added to AIGradientSuite: AIGradientSuite::IsGradientAlphaRequired. It checks whether any alpha blending is required, by checking the opacity level of each gradient stop. For details, see AIGradientSuite in *Adobe Illustrator CS4 API Reference*.

Isolation-mode improvements

Some isolation-mode restrictions were removed. The following is now possible:

- Users can enter isolation mode in most kinds of art items, such as paths, compound paths, and top-level layer.
- Users can set an insertion point on the “isolation mode” layer.

- There can be multiple art items in isolation mode; e.g., you can isolate a path/group and create another path beside it.

If your plug-in assumes these restrictions still apply, it may now behave unexpectedly.

The following changes were made to the public API.

Enum added to `AIIsolationModeType`

A new enum was added to `AIIsolationMode.h`: `AIIsolationModeType`. It defines the type of art that is isolated. The following types are defined:

- `kAIIMGroup`
- `kAIIMClippingObjects`
- `kAIIMCompoundPath`
- `kAIIMLeafArt`
- `kAIIMNone`
- `kAIIMNoObject`
- `kAIIMObjects`
- `kAIIMSubLayer`
- `kAIIMSymbol`
- `kAIIMTopLevelLayer`

For details about these new types, see the description in `IsolationMode.h`.

API added to `AIIsolationModeSuite`

A new API was added to `AIIsolationModeSuite`: `AIIsolationModeSuite::GetIsolationModeType`. It returns the type of isolation mode, an `AIIsolationModeType`. For details, see [“Enum added to `AIIsolationModeType`” on page 11](#).

OWL 2.0

In Illustrator CS3, OWL 1.0 panels were used. By enhancing the management of panels and simplifying user interactions, these provided an improved user experience. OWL 2.0 further enhances the experience and addresses some problems.

The following changes were made to the public API.

`AIScreenMode` type changes

The `AIScreenMode` was renamed from `kMultiWindowMode` to `kNormalScreenMode`.

The following `AIScreenMode` type was deprecated: `kMaximisedWindowMode`. This screen mode is now `kMaximisedWindowMode_deprecated`. If used, this type would have behaved like `kNormalScreenMode`.

For the entire list of `AIScreenMode` types and their descriptions, see `AIDocumentView.h` in the API.

ADMWindowUIMode type changes

To coincide with the changes to the AIScreenModes types, several changes were made to the ADMWindowUIMode enum.

The following ADMWindowUIMode types was removed: kADMUIMode_Maximised.

The following ADMWindowUIMode types were renamed:

- kADMUIMode_FullScreen was renamed kADMUIMode_FullScreenWithUI
- kADMUIMode_FullScreenNoMenu was renamed kADMUIMode_FullScreenWithoutUI
- kADMUIMode_Freefrom was renamed kADMUIMode_Normal

For the entire list of ADMWindowUIMode types and their descriptions, see ADMBasic.h in the API.

The following APIs are affected:

- ADMBasicSuite::GetWindowUIMode
- ADMBasicSuite::SetWindowUIMode

APIs added to ADMBasicSuite

The following new APIs were added to ADMBasicSuite:

- ADMBasicSuite::GetOpenDocumentsAsTabs
- ADMBasicSuite::SetOpenDocumentsAsTabs
- ADMBasicSuite::GetDefaultUIBrightness

The first two APIs facilitate the “setting” and “getting” of whether a new document is opened as a new tab or as a new window.

The GetDefaultUIBrightness API returns the user-interface brightness value used by Illustrator when the brightness value is not specified. This function returns an ADMUIBrightness value.

API change in ADMBasicSuite

The ADMBasicSuite::GetDocumentAreaBounds signature was changed from:

```
ADMBoolean ADMAPI (*GetDocumentAreaBounds) (ADMBoolean inInclSideDocks, ADMRect* outDimensions);
```

to:

```
ADMBoolean ADMAPI (*GetDocumentAreaBounds) (ADMRect* outDimensions);
```

kADMUIBright_default deprecated

The ADMUIBrightness value kADMUIBright_default was deprecated. To retrieve the default brightness value, use the new API, ADMBasicSuite::GetDefaultUIBrightness (see [“APIs added to ADMBasicSuite” on page 12](#)).

APIs added to ADMDialogGroupSuite

To implement floating docks (Flotillas), the following APIs were added to ADMDialogGroupSuite:

- ADMDialogGroupSuite::GetPanelsCount
- ADMDialogGroupSuite::GetNthPane
- ADMDialogGroupSuite::GetPaneName
- ADMDialogGroupSuite::GetPaneFromName
- ADMDialogGroupSuite::GetDialogGroupOrigin
- ADMDialogGroupSuite::SetDialogGroupOrigin

APIs removed from ADMDialogGroupSuite

The following API was removed from ADMDialogGroupSuite: ADMDialogGroupSuite::GetPanels. The functionality provided by ADMDialogGroupSuite::GetPanels was replaced by the new APIs listed in [“APIs added to ADMDialogGroupSuite” on page 13](#).

API changes in ADMDialogGroupSuite

The following API signatures have changed from:

- ADMErr ADMAPI (*GetDialogGroupInfo)(ADMDialogRef inDialog, const char** outGroupName, ADMInt32* outPositionCode);
- ADMErr ADMAPI (*SetDialogGroupInfo)(ADMDialogRef inDialog, const char* inGroupName, ADMInt32 inPositionCode);
- ADMBoolean ADMAPI (*IsStandAlonePalette)(ADMInt32 inPositionCode);
- ADMBoolean ADMAPI (*IsDockVisible)(ADMInt32 inPositionCode);
- ADMBoolean ADMAPI (*IsFrontTab)(ADMInt32 inPositionCode);
- ADMBoolean ADMAPI (*IsCollapsed)(ADMInt32 inPositionCode);

to:

- ADMErr ADMAPI (*GetDialogGroupInfo)(ADMDialogRef inDialog, const char** outGroupName, ADMPositionCode* outPositionCode);
- ADMErr ADMAPI (*SetDialogGroupInfo)(ADMDialogRef inDialog, const char* inGroupName, const ADMPositionCode& inPositionCode);
- ADMBoolean ADMAPI (*IsStandAlonePalette)(const ADMPositionCode& inPositionCode);
- ADMBoolean ADMAPI (*IsDockVisible)(const ADMPositionCode& inPositionCode);
- ADMBoolean ADMAPI (*IsFrontTab)(const ADMPositionCode& inPositionCode);
- ADMBoolean ADMAPI (*IsCollapsed)(const ADMPositionCode& inPositionCode);

In the above APIs, the ADMInt32 type used to contain the position code was replaced with a new type, ADMPositionCode, created specifically to hold the position-code data required to

restore a dialog's position. For details, see `ADMDialogGroupSuite` in *Adobe Illustrator CS4 API Reference*.

ADMPaneLocation type changes

The following enum was added to this type: `kADM_DPFloatingPane`. This enum identifies this pane as a floating pane.

ADMPaneState type removed

The following type was removed from `ADMPaneState`: `kADM_PanelIconLabelOnly`.

ADMDialogGroup position-code changes

The position code of the dialog group has changed slightly, with the OWL 2.0 changes. Several new bits were added to the position code used to restore a dialog's position. For a full description of the position code, see `ADMDialogGroup.h`.

Smart Guide enhancements

In CS4, Illustrator's Smart Guides feature was enhanced, to provide more information to users as they move an object around and to provide additional points in the document to snap to, including the center and edges of other artwork.

The following changes were made to the public API.

AICursorSnapSuite::Track constraint changes

Due to changes to the Smart Guides feature, two new constraints were added to the control-string parameter in `AICursorSnapSuite::Track`. The new constraints and their associated command to the snapping engine are as follows:

- L — Enables snapping to automatically generated guidelines along the center and edges of the geometric bounding box of all visible art objects.
- M — Enables snapping to automatically generated guidelines along the center and edges of all artboards in the view.

API added to AICursorSnapSuite

A new API was added to the `AICursorSnapSuite`: `AICursorSnapSuite::TrackInRect`.

This API snaps a rectangle to a specified constraint. This method supports the same constraints as `AICursorSnapSuite::Track`; however, we recommend you use only the required constraints, since this method would try snapping to all nine points (including center) of a rectangle. That could be a heavy operation, and many guides could be generated if unnecessary constraints are specified.

ATE changes

Spelling and hyphenation dictionary directories

The ATE no longer requires the paths to the spelling and hyphenation dictionaries. The following API signatures were changed from:

```
ISpell IDocumentTextResources::GetSpell(const char* pSpellingDirectory);  
void ISPELL::Initialize(const char* pSpellingDirectory);
```

to:

```
ISpell IDocumentTextResources::GetSpell();  
void ISPELL::Initialize();
```

To port to CS4, remove this parameter being passed to the API and any supporting code associated with the dictionary directory.

New AutoKernType: kMetricRomanOnlyKern

Often, roman text is mixed with Japanese text. In CS4, there is a new AutoKernType, kMetricRomanOnlyKern, which facilitates the kerning of the roman-only text, leaving the Japanese text unkerneled. This should be the default for Japanese text, allowing Mojikumi to perform its own kerning, unaffected by the kerning applied to the roman text.

Addition of SLO_COMPLEXSCRIPT APIs for Middle Eastern-specific features

New SLO_COMPLEXSCRIPT preprocessor definitions are used to define classes, types, and methods specific to Middle Eastern versions of the product. New classes defined in SLO_COMPLEXSCRIPT include the following:

- IArrayDiacVPos
- IArrayDigitSet
- IArrayDirOverride
- IArrayJustificationMethod
- IArrayKashidas
- IArrayParagraphDirection

See the API Advisor document for a full list of changes to IText.h.

New classes

Two classes were added to the ATE API:

- IArrayComposerEngine
- IArrayLine

These provide containers for Line and ComposerEngine objects, respectively. For details, see IText.h in the public API.

Menu clean-up

AIFilterSuite deprecated

As part of an effort to remove little-used menu items from the application, the AIFilter suite was deprecated in favor of AILiveEffectSuite.

If your plug-in adds a filter, it will still load in Illustrator CS4, but since there is no longer a Filter menu, your filters will be added to a Filters submenu in the Object menu. Filter support may be removed in a future release, so we recommend you change your filter to a live effect instead.

For more details, see AILiveEffectSuite in *Adobe Illustrator CS4 API Reference*. Also see the TwirlFilter sample, which adds a live effect.

Miscellaneous changes

APIs added to AIArtStyleParserSuite

The following APIs were added to AIArtStyleParserSuite:

- AIArtStyleParserSuite::CompareStyles
- AIArtStyleParserSuite::GetBlendFieldVisible
- AIArtStyleParserSuite::SetBlendFieldVisible
- AIArtStyleParserSuite::GetEffectVisible
- AIArtStyleParserSuite::SetEffectVisible
- AIArtStyleParserSuite::GetPaintBlendVisible
- AIArtStyleParserSuite::SetPaintBlendVisible
- AIArtStyleParserSuite::GetPaintFieldVisible
- AIArtStyleParserSuite::SetPaintFieldVisible
- AIArtStyleParserSuite::FindEquivalentPaintField
- AIArtStyleParserSuite::IntersectStyleWithParser
- AIArtStyleParser::GetFocusEffect
- AIArtStyleParser::SetFocusEffect
- AIArtStyleParser::SetParserFocusEffect
- AIArtStyleParser::EditEffectParametersInSelection
- AIArtStyleParser::FindEquivalentEffect
- AIArtStyleParser::StyleContainsAttributes

These APIs report and set the visibility values of the associated blends, paints, or effects; compare parsed styles for equivalence; intersect common properties between the art style and the parser; and find a particular paint field in a given parser. For details, see AIArtStyleParserSuite in *Adobe Illustrator CS4 API Reference*.

APIs added to AIDocumentSuite

The following APIs were added to AIDocumentSuite:

- AIDocumentSuite::SetDocumentBleeds
- AIDocumentSuite::GetDocumentBleeds

These new APIs were added to support the addition of bleeds to documents and facilitate the getting and setting of the bleed values.

API added to AIDocumentListSuite

A new API was added to AIDocumentListSuite: AIDocumentListSuite::AddToRecentFiles. This API adds the document referenced by a FilePath object to the recent-document list. For details, see AIDocumentListSuite in *Adobe Illustrator CS4 API Reference*.

APIs added to AIDocumentViewSuite

The following APIs were added to AIDocumentViewSuite:

- AIDocumentViewSuite::CountOPPPlates
- AIDocumentViewSuite::GetNthOPPPlate
- AIDocumentViewSuite::GetOPPPlateState
- AIDocumentViewSuite::SetOPPPlateState
- AIDocumentViewSuite::SetDocumentViewStyle
- AIDocumentViewSuite::SaveImage

AIDocumentViewSuite::SaveImage captures the entire contents of the view window.

For details about these new APIs, see AIDocumentViewSuite in *Adobe Illustrator CS4 API Reference*.

Changes to AIDrawArtSuite::DrawHilite

The AIDrawArtSuite::DrawHilite API signature has changed from:

```
AI_API AI_Error (*DrawHilite) (AI_Art_Handle art, AI_RGB_Color* color);
```

to:

```
AI_API AI_Error (*DrawHilite) (AI_Art_Handle art, AI_RGB_Color* color, float lineWidth);
```

The new parameter defines the stroke width of an art object's annotation.

Increased use of ai::UnicodeString in AIUserSuite

The AIUserSuite was modified to use ai::UnicodeString instead of ASUnicode and const char pointers to pass strings across the API. To port to CS4, change the necessary API calls to use ai::UnicodeString, and update the supporting code.

API added to AIArtSuite

The following API was added to AIArtSuite: AIArtSuite::SetKeyArt. This API replaces the deprecated AIArtSuite::CancelKeyArt API. It takes an art handle as a parameter. If that art handle

is fully selected, the API sets it as the key object for alignment, and returns `kNoErr`. If the art handle is not fully selected, `kBadParameterErr` is returned. For details, see `AIArtSuite` in *Adobe Illustrator CS4 API Reference*.

Deprecated API in `AIArtSuite`

The following API was deprecated: `AIArtSuite::CancelKeyArt`. This API is still supported in CS4, but we recommend you replace it with the new `AIArtSuite::SetKeyArt(NULL)`. For details, see `AIArtSuite` in *Adobe Illustrator CS4 API Reference*.

AutoNameGenerator changes

The name of the `_t_AIAutoNameGenerator` struct has changed to `AIAutoNameGenerator`. This struct and the `AutoNameGenerator` class were modified to use `ai::UnicodeString` instead of `ASUnicode` and C++ references instead of pointers. For details, see `AIUser.h`.

Deprecated API in `AIPathSuite`

The following API was deprecated: `AIPathSuite::CancelKeySegment`. This API is still supported in the CS4 API, but you should replace it with the new API, `AIPathSuite::SetKeySegment`. For details, see `AIPathSuite` in *Adobe Illustrator CS4 API Reference*.

API added to `AIPathSuite`

The following API was added to `AIPathSuite`: `AIPathSuite::SetKeySegment`. This API replaces the deprecated `AIPathSuite::CancelKeySegment` API. It is used to set the key anchor point. For details, see `AIPathSuite` in *Adobe Illustrator CS4 API Reference*.

AI70Color APIs removed from `AIPathStyleSuite`

The following APIs, which supported the conversion between `AIColor` objects and `AI70Color` objects, were removed:

- `AIPathStyleSuite::AIColorToAI70Color`
- `AIPathStyleSuite::AI70ColorToAIColor`

The `AI70Color` type is no longer supported. To port to CS4, you must convert any `AI70Color` types to `AIColors`.

APIs added to `AIPathStyleSuite`

The following APIs were added:

- `AIPathStyleSuite::GetCurrentPathStyleEx`
- `AIPathStyleSuite::SetCurrentPathStyleEx`
- `AIPathStyleSuite::GetPathStyleEx`
- `AIPathStyleSuite::SetPathStyleEx`

These APIs perform similar functions to `AIPathStyleSuite::GetPathStyle`, `AIPathStyleSuite::SetPathStyle`, `AIPathStyleSuite::GetCurrentPathStyle`, and `AIPathStyleSuite::SetCurrentPathStyle`, but with their extra parameters, they also return whether the fill and stroke are visible in the path style.

Change to AIUserSuite::IUAIRealToStringUnits signature

The AIUserSuite::IUAIRealToStringUnits API signature was changed from:

```
AI_API AI_Err (*IUAIRealToStringUnits) (AIReal value, ai::UnicodeString& string);
```

to:

```
AI_API AI_Err (*IUAIRealToStringUnits) (AIReal value, int precision,
ai::UnicodeString& string);
```

The new *precision* parameter defines the number of digits to the right of the decimal point. To port to CS4, add this extra parameter to any calls to IUAIRealToStringUnits. To continue using the user precision preference value, as in the prior version of this API, pass in -1 as the precision value.

Change to ADMFlashPlayerSuite::ADMFlashHostCallbackProc signature

An additional parameter, ADMItemRef inItem, was added to the ADMFlashPlayerSuite::ADMFlashHostCallbackProc callback procedure. The signature of the callback is now:

```
typedef ADMUnicode* ADM_API (*ADMFlashHostCallbackProc) (ADMItemRef inItem, const
ADMUnicode* inRequest);
```

The new parameter defines which item the callback was received for, as multiple items may register for the same callback.

APIs added to ADMDrawerSuite

The following APIs were added to the ADMDrawerSuite:

- ADMDrawerSuite::GetFontProperty
- ADMDrawerSuite::GetFontPropertyW

These new APIs return the font name and size of an ADMFont. For details, see ADMDrawerSuite in *Adobe Illustrator CS4 API Reference*.

Preference changes

The default preference value for UseLowResProxyDefault is now false. This preference sets whether a low-resolution proxy is used for linked EPS files. If your plug-in relies on the default of this preference being true, to port to CS4 you must update your plug-in to deal with this change.

There is a new preference, kShowArtboardConversionDialogKey, which sets whether the artboard legacy conversion dialog is displayed when opening a document created in a version of Illustrator earlier than CS4.

API added to AISwatchLibrariesSuite

The following API was added to the AISwatchLibrariesSuite:

- AISwatchLibraries::GetSelectedDocSwatchRefs

This API returns an array of references to selected document swatches. For details, see AISwatchLibraries.h in *Adobe Illustrator CS4 API Reference*.

Change to UTF16Traits struct

The type of the `unit_type` property in `UTF16Traits` struct was changed from `AIUTF16Char` to `ASUnicode`. Modify any supporting code that depends on the former type accordingly.

Versioned suites

- `ADMBasicSuite`
- `ADMDialogGroupSuite`
- `ADMDrawerSuite`
- `ADMFlashPlayerSuite`
- `AIArtStyleParserSuite`
- `AIArtStyleSuite`
- `AIArtSuite`
- `AICropAreaSuite`
- `AICursorSnapSuite`
- `AIDocumentListSuite`
- `AIDocumentSuite`
- `AIDocumentViewSuite`
- `AIDrawArtSuite`
- `AIFileFormatSuite`
- `AIGradientSuite`
- `AIIsolationModeSuite`
- `AIMenuSuite`
- `AIPathStyleSuite`
- `AIPathSuite`
- `AIRuntimeSuite`
- `AISwatchLibrariesSuite`
- `AIUserSuite`
- `ASLibSuite`

Legacy files removed

As the Illustrator SDK supports only a limited number of prior versions, the Illustrator 8 legacy files were removed from the SDK. For a full list of files removed, see the “Obsolete files” section of *Adobe Illustrator CS4 API Advisor*.

SDK Changes

This section summarizes changes to the organization of the SDK since the prior release.

Sample plug-in framework changes

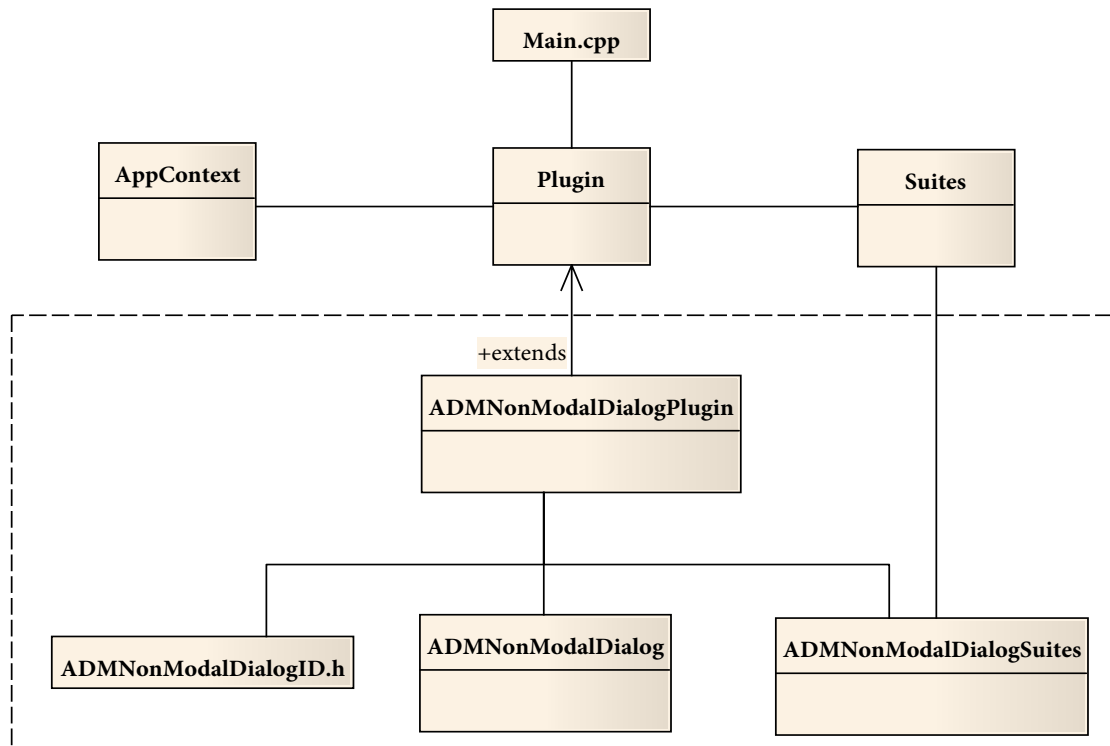
In the Illustrator CS3 SDK, there were two main frameworks used by the sample plug-ins:

- The older, C-style framework consisted of the `shellMain.cpp` file containing the plug-in entry point and several handler files implementing the plug-ins' functionality. This framework was used by `ADMNonModalDialog`, `LiveDropShadow`, `MenuPlay`, `MultiArrowTool`, `Shell`, `TextFileFormat`, `TransformButtons`, and `TwirlFilter`.
- The newer, C++-style framework was used by `MarkedObjects`, `SnippetRunner`, and `Webter`.

In the CS4 SDK, all sample plug-ins use the C++ framework, with two exceptions:

- `Shell`, which will be removed from future versions of the SDK. See [“Shell plug-in is now legacy” on page 23](#).
- The Tutorial plug-in does not use any framework. It is designed to help new users of the SDK understand the basic elements of an Illustrator plug-in without the added complications of implementing a framework.

The recommended Illustrator SDK plug-in framework is now based on a C++ class structure. All plug-ins, apart from `Shell`, were modified to use this framework. See [Figure 1](#).

FIGURE 1 C++ framework with *ADMNonModalDialog* as an example

Files and classes common to all SDK plug-ins

- **Main.cpp** — Contains the entry point to the plug-in and the `PluginMain` function, and creates the new plug-in object.
- **Plugin** class — Base class for an Illustrator SDK plug-in, this contains basic implementations of the most common functions required by a plug-in.
- **Suites** class — Acquires the suites required by the plug-in, and contains the basic suites required by the `Plugin` class.
- **AppContext** class — Uses `AIAppContextSuite` to store an application context when a notifier is received, then restore the application context after the notifier handlers are called.

ADMNonModalDialog files and classes

- **ADMNonModalDialogPlugin** class — Main plug-in class which extends the `Plugin` class and creates plug-in specific functionality and components like menu items, tools, and notifiers.
- **ADMNonModalDialog** class — Plug-in-specific class which manages the floating dialog created by this plug-in.

- `ADMNonModalDialogSuites` class — Plug-in-specific class which provides a structure containing the suites required by this plug-in. The Suites class acquires and releases these suites during plug-in reload and unload, respectively.
- `ADMNonModalDialogID.h` — Contains the resource definitions for the plug-in and is used by the project's `.r` and `.rc` files to map resource IDs to resource names.

Other classes

- `SDKAboutPluginsHelper` class — Used by SDK plug-ins to add a menu item under the About Illustrator menu item which, when selected, provides general information about the plug-in.
- `ArtMatcher` class — Provides functions to create `AIMatchingArtSpecs` and find matching art in a document using the `AIMatchingArtSuite`. This is not currently used by SDK plug-ins.
- `CASFault` class — Simple exception class which allows Illustrator error codes to be returned as exceptions. This is not currently used by SDK plug-ins.

Other files

- `IllustratorSDK.h` — Provides preprocessor defines, and includes statements for common suites and functions. This is used to create precompiled header files in Visual C++ and Xcode projects.
- `PluginStd.h` — Sets up Mac OS-specific preprocessor defines. Along with `IllustratorSDK.h`, this is used by Xcode to create the precompiled header for the project.
- `IllustratorSDKDebug.pch` and `IllustratorSDKRelease.pch` — Prefix headers for debug and release configurations, respectively. This is used by the Xcode project to generate the precompiled header from `IllustratorSDK.h` and `PluginStd.h`.
- `Pragma.h` — Used by the Visual C++ project to disable known warnings yet to be resolved.
- `SDKDef.h` — Contains version information for SDK plug-ins.

Shell plug-in is now legacy

Due to the change in the recommended framework for Illustrator plug-ins, the Shell plug-in is now a legacy plug-in. It is the only SDK sample plug-in that still uses the C-style framework. The Shell plug-in will be removed from future versions of the Illustrator SDK.

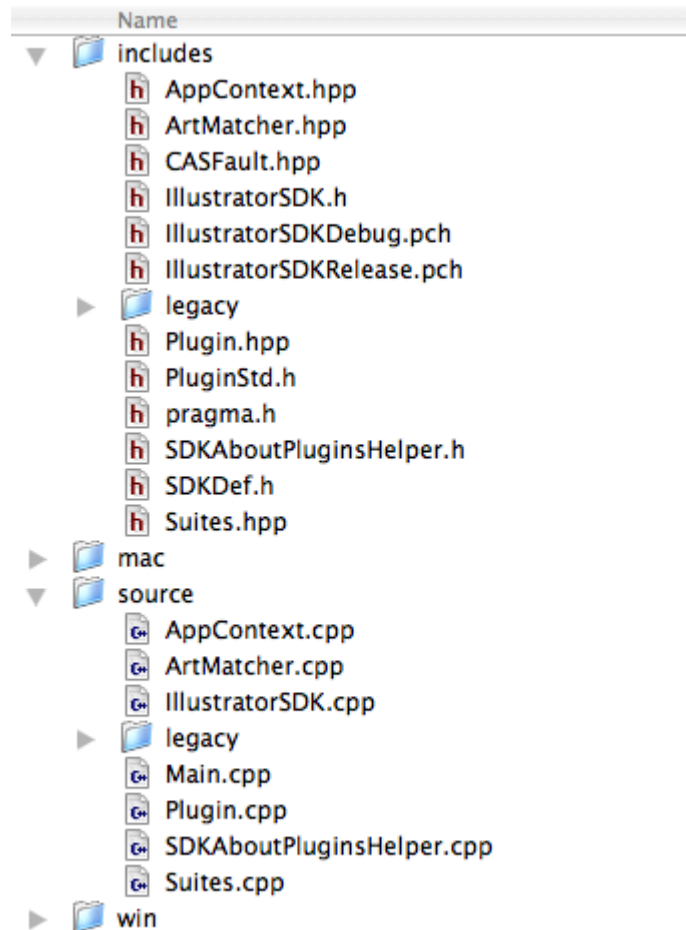
As a result of this change, the common files that are used only by the Shell plug-in were moved into a legacy folder under the `samplecode > common` directory. For details, see [“Changes to common folder structure” on page 23](#).

Changes to common folder structure

All files used by the legacy, C-style framework were moved into a legacy folder. As in the CS3 SDK, the top-level folders are `includes`, `mac`, `source`, and `win`. The difference in the CS4 SDK is that the `includes` and `source` folders now have a legacy subfolder to separate out the files used

by the legacy framework. In [Figure 2](#), the includes and source folders are expanded to show each legacy subfolder.

FIGURE 2 *Contents of the common folder*



The following common files have become legacy:

- About.h
- reportError.h
- stringUtils.h
- About.cpp
- reportError.cpp
- shellMain.cpp
- stringUtils.cpp

Removal of unused source and resource files

Several source and resource files were removed from the SDK, as they are no longer used by the sample plug-ins.

The following source and header files were removed from the project folders:

- actionManagerHandler.cpp and actionManagerHandler.h
- admHandler.cpp and admHandler.h
- Common.h
- fileFormatHandler.cpp and fileFormatHandler.h
- fileHandler.cpp and fileHandler.h
- filterHandler.cpp and filterHandler.h
- liveEffectHandler.cpp and liveEffectHandler.h
- menuHandler.cpp and menuHandler.h
- necessitiesres.h
- notifierHandler.cpp and notifierHandler.h
- Plugin.h
- plugindlg.h
- pluginGroupHandler.cpp and pluginGroupHandler.h
- pluginHandler.cpp and pluginHandler.h
- Suites.h
- timerHandler.cpp and timerHandler.h
- toolHandler.cpp and toolHandler.h
- transformAgainHandler.cpp and transformAgainHandler.h

The following source and header files were removed from the common folder:

- artset utils.cpp and artset utils.h
- file utils.cpp and file utils.h
- rasterutils.cpp and rasterutils.h
- text utils.h

Several resource files also were removed from the SDK, mostly due to the conversion of binary .rsrc files to source .r files. For details, see [“Removal of Mac OS binary resources” on page 26](#).

The following resource files were removed from the project folders:

- adm.rsrc
- filter.rsrc
- LiveDropShadowUI.rsrc
- menu.rsrc

- reportError.rsrc
- tool.rsrc
- TransformButtonsUI.rsrc
- TutorialUI.rsrc
- 257.bmp
- 257BIG.bmp
- 258.bmp
- 258BIG.bmp
- ICO00001.ico
- ICO1002.ico
- ICO1003.ico
- ICO1004.ico
- ICO1005.ico
- ICO1006.ico
- ICO1007.ico
- ICO16052.ico
- ICO16053.ico
- ICO16054.ico
- ICON1.ico
- ICON7.ico
- iconTutorial.ico
- ToolIcon.ico

The following resource files were removed from the common folder:

- about.rsrc
- ErrorStrings.rsrc
- PiPL.rsrc
- PiPL.bin

Removal of Mac OS binary resources

In the Illustrator SDK for prior versions, resources for Mac OS plug-ins were provided in binary .rsrc format. As a result, Resorcerer or ResEdit was required to view or make modifications to the resources.

In CS4, we removed these binary files and provided all Mac OS resources required by the sample plug-ins as source in .r files. Also, all resources were cleaned up, so we provide resource def-

initions for resources actually used by the plug-ins (unless the resource may be used in a future version).

Figure 3 is an example of a Mac OS resource for a dialog item. The resource types used are provided by the now obsolete Macintosh Toolbox, <http://developer.apple.com/documentation/mac/Toolbox/Toolbox-2.html>.

FIGURE 3 *Dialog resource*

```
/** Marked Objects dialog resource
 */
resource 'DLOG' (kMarkedObjectsDialogID, kMarkedObjectsDialogName, purgeable) {
    {100, 200, 370, 480}, // top, left, bottom, right
    1991, // floatZoomGrowProc
    visible, // draw on screen immediately
    goAway, // has close button
    0x0, // application options
    kMarkedObjectsDialogID, // resource ID
    kMarkedObjectsDialogName, // dialog title
    0x0 // dialog positioning noAutoCenter
};
```

Each project's resource folder now contains a .rc file containing the resource definitions for Windows and a .r file containing the resource definitions for Mac OS. Each project's source folder now also contains a <projectname>ID.h file, to map the resource name to the resource ID.

Icons used by plug-ins for dialogs or tools which were previously defined in the binary .rsrc file or as .ico files are now provided as PNG files. See Figure 4. These files are read in by the resource compiler during project build on both Windows and Mac OS.

FIGURE 4 *PNG icon resource*

```
read 'PNGI' (kMarkedObjectsToolIcons, "MarkedObjectsTool", purgeable) "MarkedObjectsTool.png";
```

NOTE: Support for old-style Mac OS icon resources (icns) will be reduced in future releases. For now, the AIAnnotatorDrawerSuite supports only 1-bit versions (ICN#). We recommend you move to PNG icons.

Cursors, being slightly more complex than icons, could not be replaced by image files. They contain extra information, like the location of the hot spot (the horizontal and vertical coordinate used to determine what the cursor is pointing), which cannot be conveyed through an image file. Instead, the cursor is defined in hex in the project's .r file. See Figure 5.

FIGURE 5 *Cursor resource*

```

/** Defines the cursor displayed when the MarkedObjects tool is selected
    and the cursor is over a document. The first 32 bytes define the cursor and
    the next 32 bytes define the mask, the last item is the h and v
    point of the hot spot.
*/
resource 'CURS' (kMarkedObjectsCursorID, locked, preload) {
    "$80 00 C0 00"
    "$E0 00 F0 00"
    "$C8 00 88 00"
    "$84 88 44 D8"
    "$42 A8 22 88"
    "$27 BE 19 AA"
    "$10 AA 08 A2"
    "$0F BE 00 00",
    "$00 00 00 00"
    "$00 00 00 00"
    "$00 00 00 00"
    "$00 00 00 00"
    "$00 00 00 00"
    "$00 00 00 00"
    "$00 00 00 00",
    {0, 0}
};

```

Building the .rsrc file

When the Xcode project is built, the resource compiler generates the project's .rsrc file from all the .r files added to the Build ResourceManager Resources build phase, plus any .r files included by these files. The resulting compiled resource is stored inside the .aip file.

Removal of Mac OS GetCursor and SetCursor functions

The Mac OS QuickDraw functions, GetCursor and SetCursor, were deprecated in Mac OS X 10.4, causing warnings to be generated during the build of sample plug-ins which used these functions. Three sample plug-ins were affected:

- MarkedObjects
- MultiArrowTool
- Shell

These plug-ins were affected because they define and use their own cursor(s) when their tool is selected from the toolbar.

In the Illustrator CS4 SDK, these plug-ins were modified to use the ADMBasicSuite member SetPlatformCursor as a cross-platform solution. For an example, see the TrackToolCursor function in MarkedObjectsPlugin.cpp.

MarkedObjects preferences dialog uses EVE resource definitions

The resource definitions for the MarkedObjects preferences dialog are now provided as EVE (Express View Engine) layout definitions. See the following new file:

`<SDK>\samplecode\MarkedObjects\Resources\MarkedObjectsPrefs.exv`.

EVE is a cross-platform solution for creating modal dialogs that are easy to maintain and correctly lay out. Used with Zstrings, EVE reduces the work required to localize dialogs.

NOTE: A limitation of EVE is that it cannot be used to define panels (or modeless dialogs).

To use EVE, follow these steps:

1. Define the resource items in text, and save in a .exv file. (For a sample EVE resource file, see `<SDK>\samplecode\MarkedObjects\Resources\MarkedObjectsPrefs.exv`.)
2. To add the resource file to a Visual Studio 2005 project (using `MarkedObjectsPrefs.exv` as an example), add the following line to the project's .rc file:

```
16500 EXVW DISCARDABLE ".\MarkedObjectsPrefs.exv"
```

3. To add the resource file to an Xcode project (using `MarkedObjectsPrefs.exv` as an example), add the following line to the project's .r file:

```
read 'EXVW' (16500, "MarkedObjects Preferences", purgeable)
"MarkedObjectsPrefs.exv";
```

The .exv file is essentially just a text file that is compiled with the project.

Dialog items are laid out by EVE dynamically, based on the text, font, groups, and alignment rules used in the dialog. Due to this dynamic rendering, exact Item positions cannot be set. We recommend not setting widths or heights in terms of absolute numbers (pixels).

EVE dialog descriptions have three primary characteristics:

- *Placement* is a property of a container that specifies how the child elements of a container are laid out, either horizontally *place_row* or vertically *place_column*. The default is vertically. A third option, *place_offscreen*, is an advanced option for designing multi-panel dialogs.
- *Spacing* is a property of a container that specifies how much spacing there should be between child elements of a container. Use *gGap* between labels and controls, *gSpace* between controls, *gLargeSpace* between clusters, and *gMargin* inside containers.
- *Alignment* is a property of a container or a child element. If specified for a container *child_horizontal/child_vertical*, it specifies how the child elements are aligned in the container. If specified for a child item *horizontal/vertical*, it specifies how the element is aligned in the container. Use *align_left/align_right* for horizontal alignment and *align_top/align_bottom* for vertical alignment. Another option, *align_center*, is for either horizontal or vertical. More advanced alignment options are *align_proportional*, which aligns items proportionally either horizontally or vertically, and *align_fill*, which expands the width or height of an item to fill a predefined space.

For more information on EVE, see the EVE language reference page on ASL (Adobe Source Libraries) — http://stlab.adobe.com/group__eve__reference.html — and the EVE tutorials

page — http://stlab.adobe.com/group__asl__tutorials__eve.html. To view the EVE widget library, go to http://stlab.adobe.com/group__widget__reference.html.

The Adobe Dialog Manager (ADM) supports EVE 1.0 (not 2.0).

Porting Case Study

This section describes the steps required to port the MarkedObjects sample from the Illustrator CS3 SDK to the Illustrator CS4 SDK. Use this as an outline of the steps needed to port your own plug-in project.

There are no known porting issues. The following are the exact steps taken to get the CS3 version of the MarkedObjects sample loaded in Illustrator CS4 on Windows and Mac OS.

Porting MarkedObjects on Windows

On a clean copy of the CS3 SDK for Windows, follow these steps

1. Navigate to the MarkedObjects folder in the CS3 SDK.
2. Double-click MarkedObjects.vcproj, to open the project in Visual Studio 2005.
3. Build the project using Build > Build MarkedObjects.
4. Select a folder in which to save the solution file.
5. Fix any compiler errors/warnings.
6. Load your plug-in in Illustrator CS4, by following the instructions in “Getting Illustrator to load your plug-in” in *Getting Started with Adobe Illustrator CS4 Development*.

Porting MarkedObjects on Mac OS

On a clean copy of the CS3 SDK for Mac OS, follow these steps:

1. Navigate to the MarkedObjects folder in the CS3 SDK.
2. Double-click MarkedObjects.xcodeproj.
3. Click Build.
4. Fix any compiler errors/warnings.
5. Load your plug-in in Illustrator CS4, by following the instructions in “Getting Illustrator to load your plug-in” in *Getting Started with Adobe Illustrator CS4 Development*.