ADOBE® INDESIGN® CS6

ADOBE INDESIGN CS6 PORTING GUIDE

Adobe

Document Update Status

CS6	All new	Content in process for CS6.

Contents

Introduction

This document describes how to update your SDK plug-in code and development environments for the Adobe® InDesign® CS6 family of applications. It details changes in the public API and other aspects of the SDK since the CS5.5 release. It contains the following:

- Chapter 1, "What's New in the InDesign CS6 Plug-In SDK," summarizes the changes in CS6 since CS5.5.
- Chapter 3, "CS6 Porting Guide," provides more details about changes to plug-in code and development environments for CS6, and helps you to port your plug-ins from CS5.5 to CS6.

Terminology

<*SDK*> refers to the installation location for the InDesign plug-in SDK.

<*Scripting SDK*> refers to the installation location for the InDesign scripting SDK.

1 What’s New in the InDesign CS6 Plug-In SDK

For the benefit of previous users, this chapter summarizes the changes to the SDK in the CS6 release:

System Requirements

Windows

- Intel® Pentium® 4 or AMD Athlon® 64 processor
- 1 GB of RAM (2 GB recommended)
- One of the following:
  - Microsoft® Windows® XP with Service Pack 2 (Service Pack 3 recommended)
  - Windows Vista® Home Premium, Business, Ultimate, or Enterprise with Service Pack 1
  - Windows 7

NOTE: Installation is not supported on volumes that use a case-sensitive file system or on removable flash-based storage devices.

Macintosh

- Multicore Intel® processor
- Mac OS® X v10.6
- 1 GB of RAM (2 GB recommended)

C++ IDE Requirements

On Windows®, the required C++ development environment is now Visual Studio 2010 with Service Pack 1 and Microsoft® Windows XP Service Pack 2 or higher.

On Mac OS®, the required C++ development environment is XCode 3.2.5 and Mac OS X 10.6.x.

Porting Content

There are some new public APIs in CS6.

See <*SDK*>/docs/references/APIAdvisor_InDesign_CS5.5_vs_CS6.html for a diff between the CS5.5 and CS6 APIs.

Liquid Layout

By using layout rules, you can create one InDesign file with one set of physical pages that display their content suitably on different devices with different sizes and orientations. You can optionally control the appearance of the pages for each size and orientation. The Liquid Layout panel is the primary way in which to interact with these features.

The InDesign CS6 scripting SDK includes the following, which demonstrate how to use related APIs:

- <*Scripting SDK*>/indesign/scriptingguide/scripts/javascript/documents/LiquidLayout.jsx
- <*Scripting SDK*>/indesign/scriptingguide/scripts/javascript/documents/AddGuides.jsx
- <*Scripting SDK*>/indesign/scriptingguide/scripts/javascript/documents/SetConstraints.jsx
- <*Scripting SDK*>/indesign/scriptingguide/scripts/vbscript/documents/LiquidLayout.vbs
- <*Scripting SDK*>/indesign/scriptingguide/scripts/vbscript/documents/AddGuides.vbs
- <*Scripting SDK*>/indesign/scriptingguide/scripts/vbscript/documents/SetConstraints.vbs
- <*Scripting SDK*>>/indesign/scriptingguide/scripts/applescript/documents/LiquidLayout.applescript
- <*Scripting SDK*>/indesign/scriptingguide/scripts/applescript/documents/AddGuides.applescript
- <*Scripting SDK*>/indesign/scriptingguide/scripts/applescript/documents/SetConstraints.applescript

Linked Content

Background: InDesign CS5.5 introduced linked stories, which made it easier to manage multiple versions of the same story or text content in the same document. You designate a story as a parent, then place the same story at other places in the document as child stories. Whenever you update the parent story, the child stories can be updated to synchronize with the parent story.

New: In InDesign CS6, you can also link simple page items, images, interactivity and groups, and so on, and you can make links from one document to another. You can manage those links and even specify and manage differences between the linked parent and child.

The InDesign CS6 plug-in SDK includes the following related content:

- *<SDK>*/source/sdksamples/codesnippets/SnpManipulateTextFrame.cpp
- *<SDK>*/source/sdksamples/codesnippets/SnpCreateAndManipulateSharedLink.cpp

Content Dropper

With this InDesign CS6 feature, you can copy the content from a file that is intended for printing and paste it into another document. With the content dropper, you can:

- Grab a lot of content at one time—including stories, groups, and natively drawn objects—directly from the page for placement elsewhere.
- Save the content that has been grabbed, so the content is retained rather than temporary as it is with the multiplace gun.
- Clearly view what has been grabbed and work with some or all of it, in any order.
- Drop the content into a different design in various ways.

The InDesign CS6 scripting SDK includes the following related content:

- *<Scripting SDK>*/ indesign/scriptingguide/scripts/javascript/documents/ContentCollectorAndPlacer.jsx
- *<Scripting SDK>*/indesign/scriptingguide/scripts/vbscript/documents/ContentCollectorAndPlacer.vbs
- *<Scripting SDK>*/ indesign/scriptingguide/scripts/applescript/documents/ContentCollectorAndPlacer.applescript

Persistent Text Fitting Option

Enabling Fit Frame to Content allows a graphic frame to automatically change size depending on its content. This option works only in basic graphical frames and text frames; it does not support more advanced text features, including mulitcolumn text frames, splits, and spans.

The InDesign CS6 scripting SDK includes the following, which demonstrate how to use related APIs:

- *<Scripting SDK>*/indesign/scriptingguide/scripts/javascript/text/PersistedTextFrameFill.jsx
- *<Scripting SDK>*/indesign/scriptingguide/scripts/vbscript/text/PersistedTextFrameFill.vbs
- *<Scripting SDK>*/indesign/scriptingguide/scripts/applescript/text/PersistedTextFrameFill.applescript

Middle Eastern Language Support

InDesign CS5 introduced Middle Eastern (ME) language support in the InDesign model and InDesign CS5.5 introduced user interface support. InDesign CS6 integrates all of the conditional code into the mainline codebase. The ME features provide support for Arabic, Hebrew, and Greek. Arabic and Hebrew are significant in that they compose right to left. In addition, CS6 adds some features, including diacritic coloring, diacritic positioning, and feature tracking.

The InDesign CS6 scripting SDK includes the following related content:

- *<Scripting SDK>*/indesignserver/scriptingguide/scripts/javascript/ChangeComposer.jsx

- <*Scripting SDK*>/indesignserver/scriptingguide/scripts/javascript/CreateStyleME.jsx
- <*Scripting SDK*>/indesignserver/scriptingguide/scripts/javascript/MECharacterAttributes.jsx
- <*Scripting SDK*>/indesignserver/scriptingguide/scripts/javascript/MENAFindAndReplaceText.jsx
- <*Scripting SDK*>/indesignserver/scriptingguide/scripts/javascript/MEParagraphAttributes.jsx
- <*Scripting SDK*>/indesignserver/scriptingguide/scripts/javascript/NumberingME.jsx
- <*Scripting SDK*>/indesignserver/scriptingguide/scripts/javascript/PageBinding.jsx
- <*Scripting SDK*>/indesignserver/scriptingguide/scripts/javascript/SpecialCharactersME.jsx
- <*Scripting SDK*>/indesignserver/scriptingguide/scripts/javascript/StoryDirection.jsx
- <*Scripting SDK*>/indesignserver/scriptingguide/scripts/javascript/TableDirection.jsx
- <*Scripting SDK*>/indesignserver/scriptingguide/scripts/vbscript/ChangeComposer.vbs
- <*Scripting SDK*>/indesignserver/scriptingguide/scripts/vbscript/CreateStyleME.vbs
- <*Scripting SDK*>/indesignserver/scriptingguide/scripts/vbscript/MECharacterAttributes.vbs
- <*Scripting SDK*>/indesignserver/scriptingguide/scripts/vbscript/MENAFindAndReplaceText.vbs
- <*Scripting SDK*>/indesignserver/scriptingguide/scripts/vbscript/MEParagraphAttributes.vbs
- <*Scripting SDK*>/indesignserver/scriptingguide/scripts/vbscript/NumberingME.vbs
- <*Scripting SDK*>/indesignserver/scriptingguide/scripts/vbscript/PageBinding.vbs
- <*Scripting SDK*>/indesignserver/scriptingguide/scripts/vbscript/SpecialCharactersME.vbs
- <*Scripting SDK*>/indesignserver/scriptingguide/scripts/vbscript/StoryDirection.vbs
- <*Scripting SDK*>/indesignserver/scriptingguide/scripts/vbscript/TableDirection.vbs
- <*Scripting SDK*>/indesignserver/scriptingguide/scripts/applescript/ChangeComposer.applescript
- <*Scripting SDK*>/indesignserver/scriptingguide/scripts/applescript/CreateStyleME.applescript
- <*Scripting SDK*>/
  indesignserver/scriptingguide/scripts/applescript/MECharacterAttributes.applescript
- <*Scripting SDK*>/
  indesignserver/scriptingguide/scripts/applescript/MENAFindAndReplaceText.applescript
- <*Scripting SDK*>/
  indesignserver/scriptingguide/scripts/applescript/MEParagraphAttributes.applescript
- <*Scripting SDK*>/indesignserver/scriptingguide/scripts/applescript/NumberingME.applescript
- <*Scripting SDK*>/indesignserver/scriptingguide/scripts/applescript/PageBinding.applescript
- <*Scripting SDK*>/indesignserver/scriptingguide/scripts/applescript/SpecialCharactersME.applescript
- <*Scripting SDK*>/indesignserver/scriptingguide/scripts/applescript/StoryDirection.applescript
- <*Scripting SDK*>/indesignserver/scriptingguide/scripts/applescript/TableDirection.applescript

Create Alternate Layout

Create Alternate Layout takes a range of existing pages with content and automatically creates linked copies of the content on a set of new pages within the same document. Use this when you need to create unique layouts for different orientations and device classes.

The InDesign Scripting SDK includes the following, which demonstrate how to use related APIs:

- <*Scripting SDK*>/indesign/scriptingguide/scripts/javascript/documents/CreateAlternateLayout.jsx
- <*Scripting SDK*>/indesign/scriptingguide/scripts/vbscript/documents/CreateAlternateLayout.vbs
- <*Scripting SDK*>/indesign/scriptingguide/scripts/applescript/documents/CreateAlternateLayout.applescript

PDF Forms Within InDesign

Historically, to create a PDF that contains form fields, you created the background in InDesign, saved the InDesign document as PDF, then opened it in Acrobat to apply form fields.

In InDesign CS6, you can apply text fields, radio buttons, signature fields, and so on directly to the InDesign document and export the document to PDF as a form with no additional Acrobat editing required, although you can edit the PDF form for additional functionality. This allows you to use InDesign's layout and formatting features to design forms.

The InDesign CS6 scripting SDK includes the following related content:

- <*Scripting SDK*>/indesign/scriptingguide/scripts/javascript/documents/ExportInteractivePDFForm.jsx
- <*Scripting SDK*>/indesign/scriptingguide/scripts/vbscript/documents/ExportInteractivePDFForm.vbs
- <*Scripting SDK*>/indesign/scriptingguide/scripts/applescript/documents/ExportInteractivePDFForm.applescript

Grayscale PDF Export

An InDesign document that contains colors can now be exported directly to a grayscale PDF, for use with newspapers or other noncolor publishers. This avoids having to maintain both color and grayscale source files for the same document.

The InDesign CS6 scripting SDK includes the following related content:

- <*Scripting SDK*>/indesign/scriptingguide/scripts/javascript/documents/GreyscalePDFforIDS.jsx
- <*Scripting SDK*>/indesign/scriptingguide/scripts/vbscript/documents/GreyscalePDFforIDS.vbs
- <*Scripting SDK*>/indesign/scriptingguide/scripts/applescript/documents/GreyscalePDFforIDS.applescript

2 Using EVE for InDesign Dialogs

After you have laid out user interface elements such as dialog boxes, changing the layout is time-consuming because, even for a small change, you might need to recalculate the positioning, sizing, and alignment of all elements. The Adobe Express View Engine (EVE) helps resolve this by resizing widgets automatically based on their text or labels. This not only makes it quick and easy to make small changes to the layout, but it also eliminates the need for multiple .fr files for different locales (in most cases) when localizing applications.

With EVE, you lay out a user interface only once and then, given a correctly formed textual description of a dialog or palette, EVE dynamically generates the geometry of the user interface elements. With EVE, you create one resource file with the relative positioning and ordering of the widgets in a row-column manner. This saves the time of generating different resources and defining the actual positions of the widgets.

You can convert your existing InDesign dialogs and panels to use the automated EVE layout format, instead of using exact widget frame coordinates. This chapter explains how to use EVE and how to convert existing user interface elements.

This chapter contains:

Prerequisites

Converting existing user interface elements to EVE is much easier if you use the EVE converter tool. The tool is located here:

```
<ID Plug-in SDK>/devtools/eveconverter/
```

Benefits of Using EVE

Laying out a dialog with EVE initially is no easier than laying it out the old way. The benefit comes when you have to modify a dialog, such as by as adding a new widget in the middle of the dialog. With EVE, you simply add the widget, and the shifting of all other widgets happens automatically.

Because you no longer need localized versions of your .fr files, there is no longer a potential problem for localized views files to have different data than the English-language .fr files. And changing an English translation that is used in a dialog does not force you to edit the .fr file to make the new text fit.

Other benefits include:

- In cases where text is different sizes on different operating systems, you no longer have to worry about calculating extra whitespace.
- English dialogs can be smaller because you do not need to leave room for anticipated localized text being longer.
- EVE eliminates problems with widgets that don’t quite line up correctly or text being chopped off because EVE automatically resizes static text, buttons, check boxes, radio buttons, and drop-down lists.

EVE Directives and Values

By default, the EVEConverter sets a regular spacing (5 pixels) between widgets with the constant kEVERegularSpaceAfter, and the overall margins to a large margin (15 pixels) on the edges with kEVELargeMargin.

The spacing defines and sizes that EVE uses are as follows:

EVE Constant	Pixel Height/Width
kEVENoSpaceAfter	0
kEVENoMargin	0
kEVESmallSpaceAfter	3
kEVERegularSpaceAfter	5
kEVESmallMargin	5
kEVELargeSpaceAfter	10
kEVERegularMargin	10
kEVEExtraLargeSpaceAfter	15
kEVELargeMargin	15
kEVEJumboSpaceAfter	20

The additional EVE content for a .fr file is defined in EVEInfo.fh in the SDK. Here’s a summary of the EVE directives:

- **Alignment**: Possible values are:
  - kEVEAlignLeft: The default for everything except buttons.
  - kEVEAlignRight
  - kEVEAlignCenter
  - kEVEAlignFill: The default for buttons. This means to take up all the space in the parent container.

  - kEVEDontAlignWidth: This means do not align to the width of the parent container.

- **Space After:** Specifies how much empty space to leave between this widget and the next one. Different widget types have different default values. Possible values are:
  - kEVENoSpaceAfter
  - kEVESmallSpaceAfter
  - kEVERegularSpaceAfter
  - kEVELargeSpaceAfter
  - kEVEExtraLargeSpaceAfter
  - kEVEJumboSpaceAfter
- **Child Arrangement**: This is the most important aspect of EVE layout. Converting a dialog mainly consists of figuring out the rows and columns in the dialog's layout. Once those are correct, the rest is just details. These values are valid only for panels. Possible values are:
  - kEVEArrangeChildrenInRow
  - kEVEArrangeChildrenInColumn
  - kEVEArrangeChildrenStacked: This means to overlay different panels or controls with different dimensions.
  - kEVEDontArrangeChildren: This means the panel widget is not to arrange the children at all. In this case, you need to specify the coordinates for each child.
- **Child Alignment**: These values are valid only for panels. They set the default alignment of children that don't provide their own alignment. Possible values are:
  - kEVEChildAlignLeft
  - kEVEChildAlignRight
  - kEVEChildAlignCenter
  - kEVEChildAlignFill
- **Child Space After**: These values are valid only for panels. They set the space after for child widgets that don't specify their own. Possible values are:
  - kEVEChildNoSpaceAfter
  - kEVEChildSmallSpaceAfter
  - kEVEChildRegularSpaceAfter
  - kEVEChildLargeSpaceAfter
  - kEVEChildExtraLargeSpaceAfter
  - kEVEChildJumboSpaceAfter
- **Margin**: Valid only for panels. This is is an inset from the edge of the panel to all the widgets. Mostly used by top level dialog widgets and group panels. Possible values are:

- ▷ kEVENoMargin
- ▷ kEVESmallMargin
- ▷ kEVERegularMargin
- ▷ kEVELargeMargin
- ▷ kEVEMinimalMargin
- ▷ kEVERegularButSmallBottomMargin
- ▷ kEVELargeButSmallBottomMargin
- ▷ kEVERegularButExtraLargeTopMargin

- **ODFRC Widget Types**: EVEGenericPanelWidget is the same as GenericPanelWidget, but adds resource fields for EVE Info. You're likely to add several of these to your dialog. There are also other widgets defined here, such as EVEStaticTextWidget.

Converting Dialogs to Use EVE

You can convert your dialogs in two ways, described in this section:

- Automatically, using the EVE conversion tool
- Manually

Converting Using the EVE Conversion Tool

NOTE: Due to an existing issue with the InDesign implementation of the EVE layout engine, some widgets may require width and height. You can get the width and height for specific widgets from the converted files that you obtain by running the tool with the -n option. This issue occurs because many of the InDesign widgets do not know how to properly size themselves, and therefore cannot report correctly to EVE. This is being addressed as the offending widgets are located.

To convert a dialog to EVE format using the conversion tool:

1. If there are additional views defined in the .fr for additional locales other than English (for example, *_deDE, *_ukUA, and so on), comment these out.
2. Run your .fr file through the EVEConverter. See "EVE Converter Command" on page 20 for command-line settings.

   If everything completes successfully, this produces a converted file with the same name as the input file and places it into the specified output directory or the default directory.

   If errors occur during processing, a dump of the widget stack is sent to STDERR and an error log file, Errors.txt, is created in the same directory as the output .fr file. You can use the -e option to automatically open the log file when the command completes.
3. Fix any errors before continuing and rerun the conversion tool. Errors may include:
   - ▷ There are several issues with dialogs containing selectable panel views. If you encounter a dialog with selectable panel views, at this time, just set its parent group's layout flag to not align the

selectable panel widget (kEVEDontAlignChildren) and leave the original FRAME directives in place.

- ▷ The most common error that comes up running the tool is an overlapping FRAME directive. In this case, usually a preprocessor directive is causing two widgets' FRAME coordinates to overlap. This is simply just a matter of finding the widget and fixing the overlapping coordinates.

  Note about the FRAME directive. Originally, this specified the bounding rectangle in exact coordinates of the widget. Under EVE, this is largely ignored. Nearly all widgets should know how to measure themselves correctly; however, there are a few special case widgets (like Tree/List based ones) that cannot at this time measure themselves. In these cases, they will still need appropriate bottom (height) and right (width) values in the FRAME directive. The other special case is when the EVE flag kEVEDontArrangeChildren is present. This causes the EVE layout engine not to attempt to arrange any child of the group which contains the flag; instead, it forces the widgets to rely on the exact coordinates specified in the FRAME directive. There should be very few cases of this flag in the future and its use should be considered highly discouraged as it defeats the purpose of having autolayout functionality.

- ▷ "Filename entered not found" error: This is usually caused by an #include statement in which the included file cannot be located. This is resolved by specifying the header search paths using the command option -p <filename.rsp>. The tool then picks the search paths from the specified .rsp file.

- ▷ "Unable to arrange in row/column" error: There may be some widgets that cannot be divided into rows and columns based on their frame coordinates due to either frames intersection or generally poorly defined coordinates in the original source (for example, miscalculated bottom right coordinates). Check the view and manually adjust the coordinates so that they no longer overlap.

- ▷ "Incorrect FRAME directive/syntax" error: This usually means that the Frame coordinates are specified either without using the "Frame" directive (that is, a macro instead of the FRAME) or Frame coordinates themselves are specified as some function or macro. In this case, adjust the Frame directives, eliminating the macro. If there are too many to do, you can run the source view file through the compiler preprocessor (Visual Studio in Windows or Xcode in MacOS) to generate a substituted version of the view, which you can then run through the converter tool.

- ▷ Dialogs that hide and show widgets. These are tricky and may require code changes, such as calls to ApplyEveLayout, to fix. Avoid using the ApplyEveLayoutToFrontDialog variant of this call, as you might inadvertently apply the EVE layout to the wrong dialog.

- ▷ Widgets are misgrouped. Sometimes the tool can put widgets together in the wrong groups; when this happens, it is fairly easy to address by moving the entire widget block to the correct group.

- ▷ If you find EVEFittedStaticTextWidget after conversion, replace it with StaticTextWidget, and provide this widget-id as border widget-id in the containing group.

- ▷ If the tool runs successfully but there is a compilation error on Windows of "Error R32741: # Error: Expression must be numeric", a possible reason could be that the widget is like an EVE generic panel or EVE cluster panel widget, which doesn't have a panel name string. Remove that and it should work fine.

4. Save a copy of your original .fr file as a backup and copy the new file in its place.

5. Run InDesign to test your converted dialog as described in "Debugging the Converted Dialog in InDesign" on page 20.

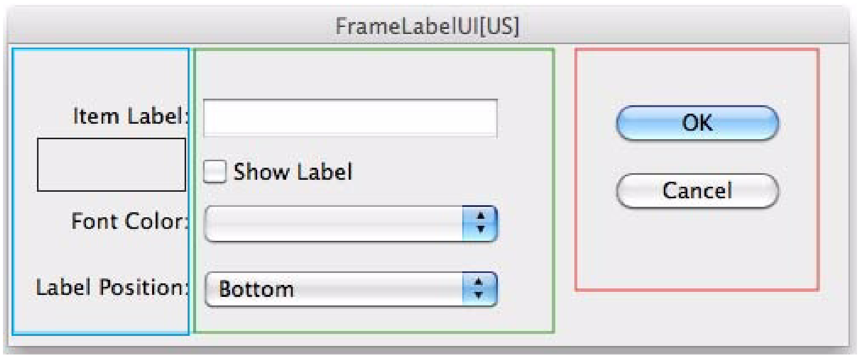
Manually Converting to EVE

For some complex dialogs, you might need to convert your dialogs manually. To do this:

1. Manually sketch out the overall layout of the dialog.
2. Decide how to divide widgets into rows and columns; see the following example, "Example 1: Dialog Layout for EVE" on page 17.
3. In the .fr file, use EVEGenericPanelWidget to divide the dialog into rows and columns.
4. Within each EVEGenericPanelWidget, embed additional EVEGenericPanelWidgets to contain buttons, checkboxes, and so on.
5. Run InDesign to test your converted dialog as described in "Debugging the Converted Dialog in InDesign" on page 20.

Example 1: Dialog Layout for EVE

Here is one example of how to assign widgets to rows and columns for EVE:



This FrameLabelUI dialog could be broken into three parts as depicted by the blue, green, and red rectangles. These three parts are in the same row, so they could be placed in a panel widget set to arrange children in a row (kEVEArrangeChildrenInRow).

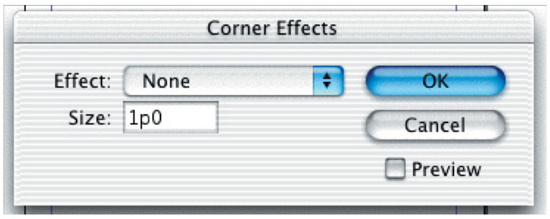
The blue rectangle contains three static text widgets vertically, where each static text widget corresponds to a widget in the green rectangle and aligns with it horizontally. The blue rectangle could be a panel widget that arranges static text widgets in a column. Note that there are four widgets in the green rectangle, but there are only three static text widgets in the blue one, so the margin between the first and second static text widgets is larger than that of other widgets. We can place an invisible static text widget (the black rectangle) there for padding purpose.

The green rectangle is similar to the blue one. You might need to specify the width of an empty text-edit box widget and a drop-down list widget with an empty selection, otherwise, its width will be very small when displayed, because EVE resizes widgets based on their text or labels.

The red rectangle is also similar to the blue one. Configuring the panel to arrange the two buttons vertically should be fine.

Example 2: Dialog Layout for EVE

Given this following original dialog, figure out which widgets should line up with what other widgets and how to arrange them in rows and columns. Note that you can lay out a dialog however you want to; this is one way to do it.



Observations about this dialog's alignments:

- Effect and Size should align on the colons.
- The drop-down list and edit box should align on their left edges.
- The buttons should align and be the same size.
- The checkbox should be centered under the buttons.

This dialog could be looked at as a single row with three columns:

- Column 1 has the static text fields (both right aligned).
- Column 2 has the drop-down list and edit box (both left aligned).
- Column 3 has the buttons and checkbox (aligned Fill and Center).

To implement this in EVE:

- To group items into a column, put them into a new panel, which arranges its children in a column. This dialog has three columns, hence three panels (EVEGe nericPanelWidgets).
- Place the dialog's three children (the three panels) into a row (kEVEArrangeChildrenInRow).
- We also want an inset all around the edge of the dialog. We'll use a large EVE margin for that (kEVELargeMargin).
- Right align the children in the first column (kEVEChildAlignRight), and put normal spacing between them (kEVEChildRegularSpaceAfter). Also use normal spacing after it, so use kEVERegularSpaceAfter.
- The second column needs a bigger space after it, so we'll use kEveExtraLargeSpaceAfter.
- The last column needs its buttons to align in the fill mode (which they do by default) and the checkbox to be centered (kEVEChildAlignCenter).
- Lastly, change the widths of the widgets that get measured. We treat the width in the .fr file as a minimum width, so if you want your dialog to become smaller when possible, you need to reduce the width of your static text, radio buttons, checkboxes, and so on.

Conversion Tips

- EVE supports automatically laying out widgets by row or column. Rows and columns can be used in combination to build complex widget layouts that should satisfy most of your needs.

  For converting any dialog, try and draw the existing dialog using coordinates for widgets. Figure out how the dialog should be divided into rows and columns. Then run the converter tool and compare the arrangement produced by the tool to your assumptions about how it should have been arranged. If the arrangement from the tool is not appropriate then you may tweak those on your own.

  Also, the tool produces many intermediate panels and dummy widgets for padding purposes. Sometimes these are not necessary and you may manually remove them. Try to use minimum numbers of dummy widgets and intermediate panels for padding.

- Don't specify coordinates for leaf widgets.

  Leave the coordinates as Frame(0,0,0,0). EVE takes care of the resizing of widgets based on their label text. However, there are two exceptions: For text edit-box widgets without an initial value and for drop-down list widgets with an empty selection, you need to specify their width; otherwise, the width will be too small.

- Fine tune your dialog

  You will need to fine tune your dialog to get it very close to the original non-EVE version; however, an exact pixel by pixel match is not usually required.

  There are several EVE properties that you could adjust to tweak your dialog:

  - Child arrangement of panel widgets: The children can be arranged in rows and columns or stacked. You can even tell the panel widget not to arrange the children at all. In this case, you need to specify the coordinates for each child.
  - Child alignment of panel widgets.
  - The margin between children and panel widgets.
  - The space between children.

  For more information on these properties, refer to EveInfo.fh in the SDK.

- If different panels or controls are overlaid with different dimensions, then use kEVEArrangeChildrenStacked. EVE evaluates the maximum size dimension out of all the overlaid panels and then, if we show or hide controls that will not show flickering, there will be no change in size. Refer to the SDK sample "snippetrunner" in the plug-in SDK (<*plug-in SDK*>/source/sdksamples/snippetrunner/SnipRun.fr).

- The tool cannot handle the following:

  - Frame coordinates that are not specified in the format Frame(left,top,right,bottom)
  - Function macros, such as #define SUM(x,y) x+y
  - In the dialog/widget definition, #include statements or conditional preprocessors, other than for WINDOWS and MACINTOSH. For example,the tool ignores #ifdef WINDOWS or #ifdef MACINTOSH. If the included file contains some useful information regarding the widgets' arrangement, then copy the contents of that file manually

Debugging the Converted Dialog in InDesign

After you have converted the dialog, test it in InDesign as follows:

1. Copy the converted file to the proper place in your source tree.
2. Open the project file associated with the .fr file and rebuild the project.
3. Launch InDesign.
4. Open the dialog for the .fr file that you converted and confirm that it looks the way you want it to.

   In most cases, the result will be close to the original, but you might need to make adjustments to spacing and alignment. To do this, close the dialog, edit the file (see the following suggestions), and rebuild even though InDesign is still running. The project won't link (on Windows), but the view resources will still be compiled and be put into the correct location.

   Tip: If your dialog doesn't look right when displayed in InDesign, you can use the panel editor to help you. It now features a display of EVE info, which might help you to figure out why your dialog is awry. It also gives you the number of pixels for space after and for margins, if you're looking for that.

5. Repeat until the dialog looks correct.

   Tip: Taking a screen shot of the dialog before conversion and at various adjustment passes can be useful in helping to determine what needs to be corrected in the layout.

Typical issues that you might encounter in this testing stage:

- InDesign crashes when you open the dialog. This is usually due to the code making assumptions about the hierarchy depth, which has now changed because of extra levels of widgets. If possible (and it is possible, although time consuming), remove assumptions like this from the code. Problems like this can also cause the dialog just not to function correctly, depending on how the code was written (checking for nil, for example).
- Clicking on some widget leads to the expansion of a dialog or the collapse of a dialog. In this case, you might have to call ApplyEveLayout() on the IEVEUtils interface, since the widget's position needs to be recomputed.
- Dialogs may start with the wrong field having focus initially. This is because the order of fields has changed in the EVE conversion and may require a code change to fix.
- Fix any issues in the code that assume the position of the widget and the widget is not found later after repositioning due to EVE conversion.
  - In most cases, there should not be source code changes needed; however, some of the more complicated dialogs may have code hooks that need to be adjusted, especially ones that attempt to walk or expect widgets to be at a certain place in the hierarchy.

EVE Converter Command

This tool converts dialogs and widgets in a .fr file to EVE notation by rearranging the widgets in a row/column manner.

The EVEConverter is a command-line tool; the command is:

```
eveconverter
```

and it is located in *<ID Plug-in SDK>*/devtools/eveconverter/.

Command-Line Options

The tool takes the following options:

Option	Description
`-i <filename>`	Required. A single input file name including the absolute path; for example: `C:\\myfolder\mycomponents\myViews.fr`
`-s <IDplugInSDKDir>`	Required. The absolute path to the SDK directory. For example: `F:\InDesign SDK\Adobe InDesign CS6 Plugin SDK\`
`-o <outputDirPath>`	Optional. Folder in which to create the output folders. The output folder `evelzed` contains the converted file and error log; the output folder `localeFiles` contains the simplified content of localized files. If not specified, the output folders are created in the folder where the input file resides. If output files with the same names already exist, they are overwritten.
`-p <searchPathsFile.rsp>`	Optional. The .rsp file that contains the search path for all header files referenced in the .fr file.
`-e`	Optional. If errors occur, this opens the error log file (Errors.txt) for viewing as soon as processing completes. The file is opened in Notepad in Windows and TextEdit in MacOS.
`-n`	Optional. Generate nonzero width and height for leaf widgets. By default, the leaf widgets have zero coordinates (height and width) and will be arranged in EVE default format.
`-x`	Optional. Avoid the grouping and row arrangement of StaticTextWidget with some related widget.

Description

Upon successful completion, the tool generates the following subfolders in the location described for the -o option:

- evelzed folder: Contains the converted file and the error log file, if any.
- localeFiles folder: Contains simplified localized files.

If there were any custom widgets defined in the .fr file, they now have an EVE prefix and are renamed in the output .fr file. For example:

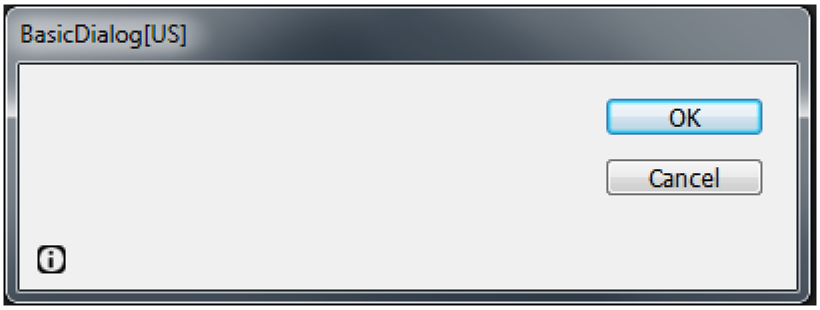
```
type BookmarkNameStaticText (kViewRsrcType) :
    StaticTextWidget(ClassID = kBookmarkNameStaticTextBoss) { };
```

becomes:

```
type EVEBookmarkNameStaticText (kViewRsrcType) :
    StaticTextWidget(ClassID = kBookmarkNameStaticTextBoss)
{
    WidgetEveInfo;
};
```

Conversion Example: basicdialog

The SDK sample `basicdialog` looks like this as delivered with the SDK:



And its description is:

```
resource BscDlgDialogBoss (kSDKDefDialogResourceID + index_enUS)
{
    __FILE__, __LINE__,
    kBscDlgDialogWidgetID,      // WidgetID
    kPMRsrcID_None,             // RsrcID
    kBindNone,                  // Binding
    Frame(0, 0, 388, 112),      // Frame (l,t,r,b)
    kTrue, kTrue,               // Visible, Enabled
    kBscDlgDialogTitleKey,      // Dialog name
    {
        DefaultButtonWidget
        (
            kOKButtonWidgetID,                        // WidgetID
            kSysButtonPMRsrcId,                       // RsrcID
            kBindNone,                                // Binding
            Frame(388-(16+80), 16, 388-16, 16+20),    // Frame (l,t,r,b)
            kTrue, kTrue,                             // Visible, Enabled
            kSDKDefOKButtonApplicationKey,            // Button text
        ),
        CancelButtonWidget
        (
            kCancelButton_WidgetID,                                  // WidgetID
            kSysButtonPMRsrcId,                                      // RsrcID
            kBindNone,                                               // Binding
            Frame(388-(16+80), 16+20+10, 388-16, 16+20+10+20),       // Frame (l,t,r,b)
            kTrue, kTrue,                                            // Visible, Enabled
            kSDKDefCancelButtonApplicationKey,                       // Button name
            kTrue,                                      // Change to Reset on option-click.
        ),
        RollOverIconButtonWidget
        (
            kBscDlgIconSuiteWidgetID,      // WidgetID
            kSDKDefIconInfoResourceID,     // Icon resource ID
            kBscDlgPluginID,               // Plug-in ID
            kBindNone,                     // Binding
            Frame(5, 87, 26, 107)          // Frame (l,t,r,b)
            kTrue, kTrue,                  // Visible, Enabled
            kADBEIconSuiteButtonDrawWellType /*kADBEIconSuiteButtonType*/,
                                           // PNGIconAttributes, new for InDesign 3.0
        ),

    },
};
```

Conversion Command

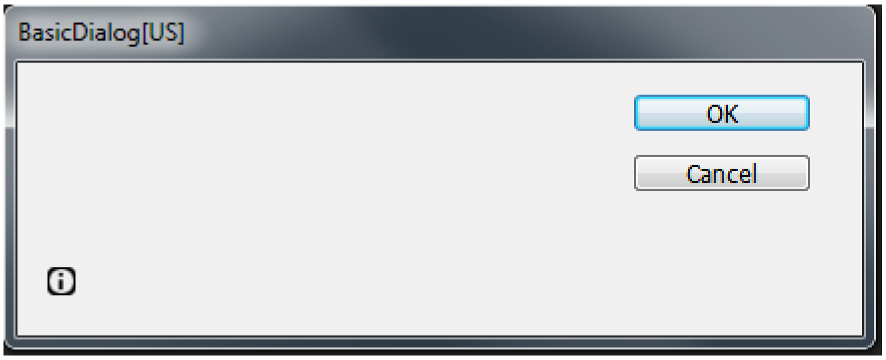
To convert this dialog to EVE format:

```
eveconverter.exe
    -i "F:\InDesign SDK\Adobe InDesign CS6 Plugin
SDK\source\sdksamples\basicdialog\BscDlg.fr"
    -s "F:\InDesign SDK\Adobe InDesign CS6 Plugin SDK"
    -p "F:\InDesignSDK\Adobe InDesign CS6 Plugin SDK\build\win\prj\SDKCPPOptions.rsp"
    -o F:\convResults
```

This processes the original BscDlg.fr file into F:\convresults\evelzed\BscDlg.fr.

Conversion Results

The converted BscDlg.fr file looks like this:



and contains the following. Text in bold is text that will be removed from the final version because the tool generated more EVEGenricPanelWidgets and dummy widgets than are actually needed.

```
resource BscDlgDialogBoss (kSDKDefDialogResourceID + index_enUS)
{
   __FILE__, __LINE__,
   kBscDlgDialogWidgetID,          // WidgetID
   kPMRsrcID_None,                 // RsrcID
   kBindNone,                      // Binding
   Frame(0,0,388,112),             // Frame (l,t,r,b)
   kTrue, kTrue,                   // Visible, Enabled
   kBscDlgDialogTitleKey,          // Dialog name
   {

       EVEGenericPanelWidget
       (
          kInvalidWidgetID,        // WidgetId
          0                        // RsrcId
          0,                       // Widget EVE Info
          kBindNone,               // Frame binding
          Frame(0,0,26,112)        // Frame
          kTrue,                   // Visible
          kTrue,                   // Enabled
          kEVEAlignFill | kEVERegularSpaceAfter | kEVEArrangeChildrenInColumn,

          {


              // Adding a dummy widget to align other widgets properly
              EVEStaticTextWidget
              (
                 kInvalidWidgetID,              // WidgetId
                 kSysStaticTextPMRsrcId,        // RsrcId
                 kBindNone,                     // Frame Binding
                 Frame(0,0,21,87)               // Frame
                 kTrue, kFalse, kAlignRight,    // Visible, Enabled, Alignment
                 kDontEllipsize, kTrue,         // Ellipsize style, Convert ampersands
                 "",
                 0 ,
                 kEVENoSpaceAfter,

              ),
              EVERollOverIconButtonWidget
              (
                 kBscDlgIconSuiteWidgetID,      // WidgetID
                 kSDKDefIconInfoResourceID,     // Icon resource ID
                 kBscDlgPluginID,               // Plug-in ID
                 kBindNone,                     // Binding
                 Frame(0,0,0,0)                 // Frame (l,t,r,b)
                 kTrue, kTrue,                  // Visible, Enabled
                 kADBEIconSuiteButtonDrawWellType /*kADBEIconSuiteButtonType*/,
                                         // PNGIconAttributes, new for InDesign 3.0

                 kEVERegularSpaceAfter,
              ),


          }  // End of Eve generic panel child widgets
       ), // End of Eve generic panel widget definition

       EVEGenericPanelWidget
       (
          kInvalidWidgetID,              // WidgetId
```

```
0                                // RsrcId
0,                               // Widget EVE Info
kBindNone,                       // Frame binding
Frame(0,0,362,112)               // Frame
kTrue,                           // Visible
kTrue,                           // Enabled
kEVEAlignFill | kEVERegularSpaceAfter | kEVEArrangeChildrenInRow,

{


   // Adding a dummy widget to align other widgets properly
   EVEStaticTextWidget
   (
      kInvalidWidgetID,             // WidgetId
      kSysStaticTextPMRsrcId,       // RsrcId
      kBindNone,                    // Frame Binding
      Frame(0,0,266,50)             // Frame
      kTrue, kFalse, kAlignRight,   // Visible, Enabled, Alignment
      kDontEllipsize, kTrue,        // Ellipsize style, Convert ampersands
      "",
      0 ,
      kEVENoSpaceAfter,

   ),

   EVEGenericPanelWidget
   (
      kInvalidWidgetID,          // WidgetId
      0                          // RsrcId
      0,
      kBindNone,                 // Frame binding
      Frame(0,0,80,50)           // Frame
      kTrue,                     // Visible
      kTrue,                     // Enabled
      kEVEAlignLeft | kEVERegularSpaceAfter | kEVEArrangeChildrenInColumn,

      {
         EVEDefaultButtonWidget
         (
            kOKButtonWidgetID,                // WidgetID
            kSysButtonPMRsrcId,               // RsrcID
            kBindNone,                        // Binding
            Frame(0,0,0,0),                   // Frame (l,t,r,b)
            kTrue, kTrue,                     // Visible, Enabled
            kSDKDefOKButtonApplicationKey,    // Button text

            kEVELargeSpaceAfter,
         ),
         EVECancelButtonWidget
         (
            kCancelButton_WidgetID,               // WidgetID
            kSysButtonPMRsrcId,                   // RsrcID
            kBindNone,                            // Binding
            Frame(0,0,0,0),                       // Frame (l,t,r,b)
            kTrue, kTrue,                         // Visible, Enabled
            kSDKDefCancelButtonApplicationKey,    // Button name
```

```
                        kTrue,                      // Change to Reset on option-click.

                        kEVELargeSpaceAfter,
                    ),


                }  // End of EVE Generic panel child widgets
            ), // End of EVE Generic panel widget definition

        }  // End of Eve generic panel child widgets
    ), // End of Eve generic panel widget definition
},

kEVEArrangeChildrenInRow | kEVELargeMargin,
};
```

Adjusting the Result

You might or might not want or need your dialog to match the original pixel by pixel. However, if you want it to be closer to the original than might occur with the generated dialog, you can adjust the results manually.

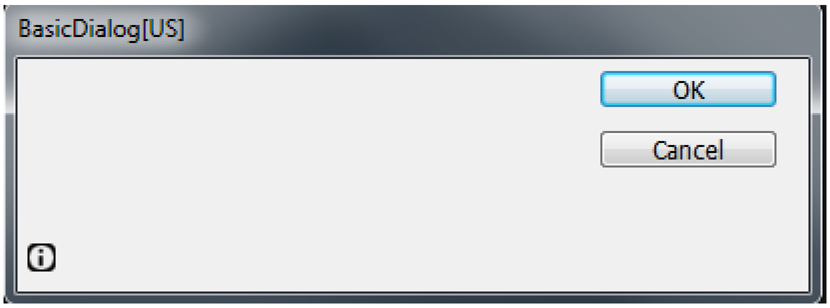
First of all, we check whether we can remove any unnecessary EVEGenericPanelWidget and dummy widgets. For this example, the tool does not generate optimum results, so we remove the unnecessary EVEGenericPanelWidget and the dummy widget manually as shown by the bold text in the preceding Conversion Result.

For this example, we also need to adjust the coordinate of the remaining EVEGenericPanelWidget to divide the dialog. This is not necessary if you do not remove any EVEGenericPanelWidget.

Because EVE lays out widgets by row and column groupings, you should first focus on adjusting any alignment (especially static text labels and such), then move on to cosmetic spacing.

The large margin is too big, so we change that to kEVESmallMargin (5 pixels), which gets us closer in height. By default, the EVEGenericPanelWidget is given regular spacings that cause the OK/Cancel button to not stay in the top-right corner as in the original dialog, so, for the right EVEGenericPanelWidget, we adjust its spacing to large (10 pixels).

This results in the final dialog:



Changes from the generated file are shown in bold:

```
resource BscDlgDialogBoss (kSDKDefDialogResourceID + index_enUS)
{
    __FILE__, __LINE__,
    kBscDlgDialogWidgetID,          // WidgetID
    kPMRsrcID_None,                 // RsrcID
    kBindNone,                      // Binding
    Frame(0,0,388,112)              // Frame (l,t,r,b)
    kTrue, kTrue,                   // Visible, Enabled
    kBscDlgDialogTitleKey,          // Dialog name
    {

        EVEGenericPanelWidget
        (
            kInvalidWidgetID,       // WidgetId
            0                       // RsrcId
            0,                      // Widget EVE Info
            kBindNone,              // Frame binding
            Frame(0,0,280,112)      // Frame
            kTrue,                  // Visible
            kTrue,                  // Enabled
            kEVEAlignFill | kEVERegularSpaceAfter | kEVEArrangeChildrenInColumn,

            {
                // Adding a dummy widget to align other widgets properly
                EVEStaticTextWidget
                (
                    kInvalidWidgetID,               // WidgetId
                    kSysStaticTextPMRsrcId,         // RsrcId
                    kBindNone,                      // Frame Binding
                    Frame(0,0,280,87)
                    kTrue, kFalse, kAlignRight,     // Visible, Enabled, Alignment
                    kDontEllipsize, kTrue,          // Ellipsize style, Convert ampersands
                    "",
                    0 ,
                    kEVENoSpaceAfter,

                ),
                EVERollOverIconButtonWidget
                (
                    kBscDlgIconSuiteWidgetID,       // WidgetID
                    kSDKDefIconInfoResourceID,      // Icon resource ID
                    kBscDlgPluginID,                // Plug-in ID
                    kBindNone,                      // Binding
                    Frame(0,0,0,0)                  // Frame (l,t,r,b)
                    kTrue, kTrue,                   // Visible, Enabled
                    kADBEIconSuiteButtonDrawWellType /*kADBEIconSuiteButtonType*/,
                                                // PNGIconAttributes, new for InDesign 3.0

                    kEVERegularSpaceAfter,
                ),
            }  // End of Eve generic panel child widgets
        ), // End of Eve generic panel widget definition


        EVEGenericPanelWidget
        (
            kInvalidWidgetID,               // WidgetId
            0                               // RsrcId
            0,
            kBindNone,                      // Frame binding
            Frame(0,0,70,50)                // Frame
            kTrue,                          // Visible
            kTrue,                          // Enabled
            kEVEAlignLeft | kEVELargeSpaceAfter | kEVEArrangeChildrenInColumn,

            {
               EVEDefaultButtonWidget
               (
                  kOKButtonWidgetID,                // WidgetID
                  kSysButtonPMRsrcId,               // RsrcID
                  kBindNone,                        // Binding
                  Frame(0,0,0,0)                    // Frame (l,t,r,b)
                  kTrue, kTrue,                     // Visible, Enabled
                  kSDKDefOKButtonApplicationKey,    // Button text

                  kEVELargeSpaceAfter,
               ),
               EVECancelButtonWidget
               (
                  kCancelButton_WidgetID,                   // WidgetID
                  kSysButtonPMRsrcId,                       // RsrcID
                  kBindNone,                                // Binding
                  Frame(0,0,0,0)                        // Frame (l,t,r,b)
                  kTrue, kTrue,                         // Visible, Enabled
                  kSDKDefCancelButtonApplicationKey,    // Button name
                  kTrue,                            // Change to Reset on option-click.

                  kEVELargeSpaceAfter,
               ),
            }  // End of EVE Generic panel child widgets
        ), // End of EVE Generic panel widget definition

    },

    kEVEArrangeChildrenInRow | kEVESmallMargin,
};
```

Conversion Example: Selectable Dialogs

Another example is BasicSelectableDialog. In this example, there are list and tab style selectable dialogs derived from kSelectableDialogBoss, and two panel widgets derived from PrimaryResourcePanelWidget.

As in the previous example, you can run the EVE converter to generate a converted .fr and adjust the result manually to remove any unnecessary EVEGenericPanelWidget and dummy widgets; adjust the coordinate of the EVEGenericPanelWidget or child widget; adjust the alignment, margins, and spacing; and so on.

However, in this example, the EVE changes for the tree node widget are not required because this widget is created by the code and typically it is not passed to the EVE engine. Therefore, we need to revert any changes to it after the conversion.

After the conversion, the .fr looks like this:

```
type BscSlDlgTreeNodeWidget (kViewRsrcType) : PrimaryResourcePanelWidget (ClassID =
kTreeNodeWidgetBoss){
    WidgetEveInfo;
};

resource BscSlDlgTreeNodeWidget (kBscSlDlgTreeNodeRsrcID)
{
    __FILE__, __LINE__,
    kBscSlDlgTreeNodeWidgetID, kPMRsrcID_None,      // WidgetId, RsrcId
    kBindLeft | kBindRight,
    Frame(0,0,149,18),                              // Frame
    kTrue, kTrue,                                   // Visible, Enabled
    "",                                             // Panel name
    {
        EVEInfoStaticTextWidget
        (
            kBscSlDlgTreeNodeNameWidgetID,
            kSysStaticTextPMRsrcId,             // WidgetId, RsrcId
            kBindNone,                          // Frame binding
            Frame(3,1,120,17)                   // Frame
            kTrue, kTrue, kAlignLeft,           // Visible, Enabled, Alignment
            kDontEllipsize, kTrue,              // Ellipsize style, Convert ampersands
            "",
            0,
            kPaletteWindowSystemScriptFontId, kPaletteWindowSystemScriptHiliteFontId,

            kEVERegularSpaceAfter,
        ),
    }

    kEVEArrangeChildrenInRow | kEVELargeMargin,
}
```

We then revert the changes:

```
type BscSlDlgTreeNodeWidget (kViewRsrcType) : PrimaryResourcePanelWidget (ClassID =
kTreeNodeWidgetBoss){
};
resource BscSlDlgTreeNodeWidget (kBscSlDlgTreeNodeRsrcID)
{
    __FILE__, __LINE__,
    kBscSlDlgTreeNodeWidgetID, kPMRsrcID_None,     // WidgetId, RsrcId
    kBindLeft | kBindRight,
    Frame(0,0,149,18),                             // Frame
    kTrue, kTrue,                                  // Visible, Enabled
    "",                                            // Panel name
    {
        InfoStaticTextWidget
        (
            kBscSlDlgTreeNodeNameWidgetID,
            kSysStaticTextPMRsrcId,            // WidgetId, RsrcId
            kBindNone,                         // Frame binding
            Frame(3,1,120,17)                  // Frame
            kTrue, kTrue, kAlignLeft,          // Visible, Enabled, Alignment
            kDontEllipsize, kTrue,             // Ellipsize style, Convert ampersands
            "",
            0,
            kPaletteWindowSystemScriptFontId, kPaletteWindowSystemScriptHiliteFontId,
        ),
    }
}
```

Additional Examples

Some other SDK samples have been converted to EVE/Drover style in the plug-in SDK, including:

- BasicDialog
- BasicLocalization
- BasicPersistInterfaceUI
- BasicSelectableDialog
- FrameLabelUI
- SnippetRunner

You can get them from <*SDK*>/source/sdksamples/.

3 CS6 Porting Guide

This chapter describes changes to plug-in code and development environments caused by changes in the public API and other aspects of the SDK for the InDesign CS6 family of applications since CS5.5, and helps developers port their plug-ins to CS6 from CS5.5.

Main Changes

New ODFRC Compile Error

The new ODFRC.exe checks whether .fr files that are marked as kResourceUTF8Encoded contain valid UTF8 characters.

There a question whether our .fr files with kResourceUTF8Encoded should be saved with the UTF8 signature. Old xcode did not work with UTF8 signatures, so it was not an option before. Now we could, but Unicode does not recommend saving files with the UTF8 signature. They recommend that applications assume that a file is UTF8 unless it encounters invalid UTF8.

Unfortunately, Visual Studio does not do this. They assume that a file is code page 1252 unless it has UTF8 characters. This means that, if you open a string file with no UTF8 in Visual Studio and add an accented character (or any character that exists in 1252), the file will be saved by default in 1252.

To fix the compile error, you can use “File->Advanced Save Options” and choose “Unicode (UTF-8 without signature) – Codepage 65001”.

Porting Recipes

This section includes a handful of changes that you will likely run into right away. This may help you to quickly get beyond the most common compilation errors.

CControlView

Rename the following:

- **CreateTransform** to CreateViewTransform
- **DestroyTransform** to DestroyViewTransform

CGraphicFrameShape

GetPaintedBBox: The second argument is now const. This should never have been changeable. If your code changes this CoordinateSpace, it is in error. See CPathShape.

CGraphicPlaceBehavior

ProcessDoPlaceInfo, ProcessPlace, ProcessReplace, and ProcessReplaceMe: An additional optional parameter is added to specify whether to use the place gun or the content dropper to place content. By default, it uses the place gun and behaves the same as in CS5.5.

CHandleShape

HitTestOneHandle: Add the required parameter PointIndex, which improves the page tool selection.

CPMDataObject

Rename the following:

- **GetScrapRef** to GetPasteboardRef
- **SetScrapRef** to SetPasteboardRef

CPanorama

Rename the following:

- **Scale** to ScalePanorama
- **ScaleBy** to ScalePanoramaBy

GetXScaleFactor and **GetYScaleFactor**: A new optional bool parameter is available. Pass kTrue to get the zoom level that should be displayed to the user. By default, InDesign adjusts the layout view zoom factor to account for the resolution of the monitor that the panorama is displayed within such that, when viewed at what the user believes is 100%, one inch of the ruler will take up one inch of monitor space. If you do not want this monitor scale factor included, then pass kFalse. The default is kTrue, which behaves the same as in CS5.5.

CPathShape

CanMakeShapePath: Remove.

ShapePath: Remove.

GetPaintedBBox: The second argument is now const. This should never have been changeable. If your code changes this CoordinateSpace, it is in error. See CShape.

CShape

CanMakeShapePath: Remove.

ShapePath: Remove.

GetPaintedBBox: The second argument is now const. This should never have been changeable. If your code changes this CoordinateSpace, it is in error.

CTUnicodeTranslator

Change parameter type from long to int32 in the following:

- **CharToTextChar**
- **TextCharToChar**
- **TextCharToChar_Exact**

CXMLOutStream

ResetEntityMap: A new optional bool parameter is available to specify whether to add a standard XML Entity or not. By default it is kTrue and behaves the same as in CS5.5.

Façade::IFrameContentFacade

ErrorCode: Change the return type to bool16.

Façade::IGeometryFacade

AddResizeActionAtom, CanChangeItemsHeight, CanChangeItemsLength, CanChangeItemsWidth, ConstructResizeScriptMethod, GetItemBounds, GetItemsDimensions, GetItemsHeight, GetItemsLength, GetItemsSize, GetItemsWidth, ItemsHaveHeight, ItemsHaveWidth, RecordResize, ResizeItems, and **SetItemsBounds**: The second argument is now const. This should never have been changeable. If your code changes this CoordinateSpace, it is in error.

Façade::ISharedContentFacade

Instead of using a UID list to create shared content links, use source and target UID references.

ContainsSharedContentLink: Rename to ContainsSharedStoryLink.

IsSharedContentLink: Replace with IsSharedContentInternalLink.

Added other methods to check the type of a content link, such as IsSharedContentExternalLink, IsSharedStoryLink, and IsSharedPageItemLink.

Façade::ITransformFacade

AddTransformActionAtom, CanChangeItemsTransformValues, CanRemoveItemsTransformations, ConstructTransformScriptMethod, GetItemsRotationAngle, GetItemsScale, GetItemsScale, GetItemsSkewAngle, GetItemsTransform, GetItemsTranslation, GetItemsTranslation, ItemsHaveTransformValues, RecordTransform, RemoveItemsTransformations, TransformItems, and **TransformPathPoints**: The second argument is now const. This should never have been changeable. If your code changes this CoordinateSpace, it is in error.

Geometry::ResizeValue

ResizeValue: The last argument is now const. This should never have been changeable. If your code changes this CoordinateSpace, it is in error.

IAutoFlowCmdData

Set: A new optional argument is available to specify whether to use the place gun or the content dropper.

ICellContent

NotifyRowTypeChanging: Change the first argument's type from rowType to bool16. Cells support two "types":

- `regular`: For each Layout Row, a Parcel is created and the text flows into it as if it were linked.
- `repeat`: In which the text composes only in the first Parcel and all the other Parcels simply draw the contents of the first Parcel. This is used to support Header/Footer and SubHeader. This does not mean that a cell "is a header or footer"; rather, this just indicates what its type is.

IClipboardController

GetScrapRef: Remove.

IDFile

GetFSSpecName: Remove.

IDocSetupCmdData

PMRect type: Replace with the new type, PMPageSize.

GetPageSize: Rename to GetDocumentPageSize and change its return type from PMRect to PMPageSize.

Set: Rename to SetDocSetupCmdData and change the type of the second argument from PMRect to PMPageSize.

SetPageSize: Remove.

IDocumentPresentation

PaletteRef: Remove.

GetSelectableViews: Change the argument type to K2Vector<IContorlView*>.

IDocumentUIState

Rename the following:

- **GetScaleFactor** to GetScaleFactor_

- **SetScaleFactor** to SetScaleFactor_

IExportTagValues

GetTagData and **SetTagData**: The first argument is now const. This should never have been changeable. If your code changes this WideString, it is in error.

IFormFieldSuite

DestroyFieldFromSelection: A new optional argument named formFieldToConvert is available. If this method is invoked in the course of converting some form fields to another type, then this parameter would be the destination type for the form field.

IFramePrefsCmdData

Set: Add the following required parameters:

- useFlexibleWidth: If true, use flexible column widths.
- maxColumnWidth: Maximum column width; used when useFlexibleWidth is kTrue.
- asDimension: Which dimension to use for autosizing. Valid values are Off, Width, Height, Both, or proportionally.
- refPoint: Autosizing reference point from 9-point proxy.
- useMinHeight: If true, use minHeightValue for autosizing.
- minHeightValue: The minimum height for autosizing.
- useMinWidth: If true, use minWidthValue for autosizing.
- minWidthValue: The minimum width for autosizing.
- noLineBreak: If true, line breaks are not applied in autosizing.

IFullScreenRestoreData

GetRestoreRect: Rename to GetRestoreMargins and change its return type from "const PMRect&" to "PMRect".

SetRestoreRect: Rename to SetRestoreMargins.

IGenericSettingsExportCmdData

Remove the following:

- **GetFileCreator**
- **GetFileType**
- **SetFileCreator**
- **SetFileType**

IGeometrySuite

Change all CoordinateSpace parameters to const in the following:

- **CanChangeSelectionHeight, CanChangeSelectionLength, CanChangeSelectionSize, CanChangeSelectionWidth, GetSelectionBounds, GetSelectionHeight, GetSelectionLength, GetSelectionSize, GetSelectionWidth, ResizeSelection, SelectionHasHeight, SelectionHasSize**, and **SelectionHasWidth**.

IGlyphUtils

Change the parameters' type and return value from long to int32 in the following:

- **GetGlyphAccessInfo, GetGlyphHasSfntAlternate, GetGlyphsForFeatureAccessInfo, GetOTFAttribute**, and **TestFeatureSubstitution**.

IGlyphUtilsME

Change the parameters' type and return value from long to int32 in the following:

GetGlyphAccessInfo, GetGlyphsForFeatureAccessInfo, GetOTFAttribute, and **TestFeatureSubstitution**.

IGuideDataSuite

ChangeGuideAttributes: Remove.

GetViewThreshold: Change argument's name to thresholdList.

IGuideUIUtils

MakeChangeGuideColorCmd: Remove.

IInCopyFileInfo

PageRect and **GetPrintedBoundingBox**: Change argument's type from PMRect to PMPageSize.

IJustificationStyle

SetJustificationMethod: Rename the argument to justMethod.

ILayoutControlData

AttachDocObservers: Change the argument's type to IDocument.

AttachWorkspaceObservers: The argument is now const. This should never have been changeable. If your code changes this IDocument, it is in error.

ILayoutUIUtils

GetNextZoomFactor: A new optional argument named adjustForMonitorPPI is available. This argument is the factor in an additional scale to adjust for the resolution of the monitor that is displaying this view.

GetNextZoomFactor and **GetNextZoomValue**: Change the second argument from bool16 to bool32.

ILink

InitFromINX: A new optional argument named lastExportTime is available. This indicates whether the link is a shared content link.

ILinksUIUtils

GetLinkResourcesSelectedInPanel, GetLinksForMenuAction, and **GetLinksSelectedInPanel**: A new optional argument named inDisplayOrder is available. This argument means to operate the links in the order of their display in the links panel.

IMenuManager

HandlePopupMenu: A new optional argument named excludeRect is available. If this argument is set, pop-up menus should not overlap the coordinate rectangle contained in excludeRect.

IMoveReferencePointCmdData

Rename the following:

- **GetNewReferenceFrame** to GetNewContentOffSetFrame
- **GetNewPosition** to GetNewReferencePointLocation
- **GetOldPosition** to GetOldReferencePointLocation
- **SetOldLocation** to SetOldReferencePointLocation
- **GetLayoutData** to GetReferencePointLayoutData

Remove the following:

- **GetRefPointBoss**
- **SetOldReferenceFrame**
- **SetRerenceFrame**

IMultiColumnItemData

Rename the following:

- **GetPageSizePrefto** GetNewDocumentPageSize
- **SetPageSizePref** to SetNewDocumentPageSize

GetXMLInterchangeDocumentVersion: Remove

INewGuideCmdData

Set: A new optional argument named guideType is available. This argument specifies which type of guide to create (for example, ruler, liquid, magnetic, and so on).

INewSpreadCmdData

SetNewSpredCmdData: Change from a virtual function to an inline function.

GetNewSpredIsIslandSpread: Remove.

INoteUIUtils

DoDragDropNote: Remove.

IObjectExportOptions

CSSUnitType: Replace with CustomLayoutType when customizing alignment and space.

IObjectExportOptionsData

CSSUnitType: Replace with CustomLayoutType when customizing alignment and space.

IObjectExportOptionsSuite

CSSUnitType: Replace with CustomLayoutType when customizing alignment and space.

IOpenLayoutPresentationCmdData

Rename the following:

- **GetPerspective** to GetPerspective_
- **SetPerspective** to SetPerspective_

IPMDataObject

Remove the following:

- **GetScrapRef**
- **SetScrapRef**

IPageCmdData

Set:

- Rename to SetNewPageCmdData
- A new optional argument named pageSize is available to set the page size. Valid values are:
  - kPMPageSizeNeighbor, the default, which means to use the page size of the location where the spread is being inserted.
  - kPMPageSizeDefault, which means use the document's default page size.

IPageSetupPrefs

GetPageSizePref: Change the type of the return value from PMRect to PMPageSize.

SetPageSizePref: Change the type of the argument from PMRect to PMPageSize.

IPageSizes

GetNthPageSize: Remove the second argument and change the return type to PMPageSize.

IPagesPanelPrefs

ViewSetting: Introduced options to set view. There are four options:

- kInvalidView: Don't change the view setting.
- kHorizontally: Standard setting for rows of pages arranged from left to right, multiple pages per row.
- kVertically: One page per row, arranged and stacked vertically.
- kByAlternateLayout: Each alternate layout is arranged in a vertical stack.

Added two methods, SetViewSetting and GetViewSetting.

SetVerticalView(kTrue): Replace with SetViewSetting(kHorizontally).

IPagesPanelPrefsCmdData

Use **ViewSetting** option. See IPagesPanelPrefs.

IPanorama

Rename the following:

- **Scale** to ScalePanorama
- **ScaleBy** to ScalePanoramaBy

IProxyWidgetAttributes

Rename the following:

- **IsLine** to GetProxyIsLine

- **SetIsLine** to SetProxyIsLine
- **SetReferenceAngle** to SetProxyReferenceAngle
- **SetReferenceXScale** to SetProxyReferenceXScale
- **SetReferenceYScale** to SetProxyReferenceYScale
- **SetReferenceXSkew** to SetProxyReferenceXSkew
- **SetReferenceLine** to SetProxyReferenceLine
- **GetReferenceAngle** to GetProxyReferenceAngle
- **GetReferenceXScale** to GetProxyReferenceXScale
- **GetReferenceYScale** to GetProxyReferenceYScale
- **GetReferenceXSkew** to GetProxyReferenceXSkew
- **GetReferenceLine** to GetProxyReferenceLine
- **SetAppearance** to SetProxyAppearance
- **GetAppearance** to GetProxyAppearance
- **SetPosition** to SetProxyPosition
- **SetProxyPosition** to GetProxyPosition
- **UpdateState** to UpdateProxyState
- **UpdateValues** to UpdateProxyValues
- **UpdateRotation** to UpdateProxyRotation
- **UpdateScale** to UpdateProxyScale
- **UpdateSkew** to UpdateProxySkew

IRectListData

Set: Rename to SetRectList.

IRefPointUIUtils

GetActiveReferencePoint: Remove.

GetReferencePoint: Rename to GetPasteboardReferencePoint.

IRefPointUtils

Rename the following:

- **CalculateReferencePoint** to CalculateReferencePoint_1
- **CalculateReferencePoint** (the other overloaded method) to CalculateReferencePoint_2

IReferencePointData

Make all methods private and rename the following:

- **SetShown** to SetShown__
- **IsShown** to IsShown__
- **SetPosition** to SetReferencePointData
- **GetPoint** to GetPosition__
- **SetRefPointLocked** to SetRefPointLocked__
- **IsRefPointLocked** to IsRefPointLocked__
- **SetVisible** to SetVisible__
- **IsVisible** to IsVisible__
- **GetOffsetFromCenter** to GetOffsetFromCenter__
- **SetOffsetFromCenter** to SetOffsetFromCenter__
- **GetReferenceFrame** to GetReferenceFrame__
- **SetReferenceFrame** to SetReferenceFrame__

Remove the following:

- **SetPoint**
- **SetAngle**
- **GetAngle**

Rename the enums in **ReferencePointPosition** that describe the reference point position:

- kTopLeft to kTopLeftReferencePoint
- kFirstPosition to kFirstReferencePointPosition
- kTopCenter to kTopCenterReferencePoint
- kTopRight to kTopRightReferencePoint
- kLeftCenter to kLeftCenterReferencePoint
- kCenter to kCenterReferencePoint
- kRightCenter to kRightCenterReferencePoint
- kBottomLeft to kBottomLeftReferencePoint
- kBottomCenter to kBottomCenterReferencePoint
- kBottomRight to kBottomRightReferencePoint
- kLastPosition to kLastReferencePointPosition

- kOther to kOtherReferencePointPosition
- kNone to kUninitializedReferencePointPosition

IReferencePointSuite

Add the new required argument with type of IControlView to the following:

- **ViewHasReferencePoint, GetViewDefaultReferencePosition, GetViewReferencePointPosition, GetViewReferenceAppearance, GetViewReferenceLine, CanChangeViewReferencePoint, CalculateViewReferencePoint, ChangeViewReferencePoint**, and **ChangeViewReferencePosition**.

ISaveFileDialog

AddFileTypeInfo: Remove two arguments named creatorType and fileType.

SetClientName: Remove.

ISectionList

QueryOrderedList: Remove the argument.

ISelectionFilter

FilterDeselect and **FilterSelection**: Add the new required argument with type IConcreteSelection. This argument is the selection boss.

ISetAdobeMediaMgmtMDCmdData

Set: Add the new required argument named origDocumentID. This argument means the common identifier for the original document.

ISetFittingOptionsCmdData

Set: Rename to SetFittingOptionsData.

IShape

CanMakeShapePath: Remove.

GetPaintedBBox: The second argument is now const. This should never have been changeable. If your code changes this CoordinateSpace, it is in error.

ISharedContentSuite

CreateSharedContentLink: A new optional argument named targetDB is available. If this argument is set to nil, then create a shared content link within the same document. If this argument is not nil, then create a shared content link across documents.

ISpread

GetPagesAndItemsBounds and **GetPagesBounds**: The first argument is now const. This should never have been changeable. If your code changes this CoordinateSpace, it is in error.

ISummaryUtils

ExportHumanTextSummary: Remove two arguments named typeCreator and typeFile.

ITextParcelList

Recompose:

- Remove the argument named fitContainerToContent.
- Optionally add the new argument with type of PMReal.

ITinDocumentData

Remove the following:

- **AddGaiji, CacheGlyphletsInStory, ClearCachedGaijis, GatherGlyphletBlobData, GetDocumentTinID, GetFontsWithGaiji, GetGlyphletBlob, GetGlyphletBlobLength, HasCachedGlyphlets, HasGlyphletPackage, ModifyGaiji, ReadWriteCachedGlyphletPackage, RegisterGaijiData, RemoveGaiji, SetDocumentTinID**, and **SetGlyphletBlob**.

ITransformPanelPrefs

GetShowGlobal and **SetShowGlobal**: Remove; use GetCoordinateSpace and SetCoordinateSpace instead.

ITransformPanelPrefsCmdData

GetChangeShowGlobal, **GetShowGlobal,** and **SetShowGlobal**: Remove; use GetChangeCoordinateSpace, GetCoordinateSpace, and SetCoordinateSpace instead.

ITransformSuite

CanChangeSelectionTransformValues, CanRemoveSelectionTransformations, GetSelectionRotationAngle, GetSelectionScale, GetSelectionScaleX, GetSelectionScaleY, GetSelectionSkewAngle, GetSelectionTranslation, GetSelectionTranslationX, GetSelectionTranslationY, RemoveSelectionTransformations, SelectionHasTransformValues, and **TransformSelection**: The first argument is now const. This should never have been changeable. If your code changes this CoordinateSpace, it is in error.

IWaxAnchorPt

MarkKeepsDamage: Remove.

IWidgetUtils

IsTextLeftToRight: Remove.

IsFileInTrash: Change the argument type from FSSpec to FSRef.

IWindowUtils

SnapRectToNearestPalette: Remove.

MColorContext

IsMonitorColor: Remove.

PMString

Remove the following:

- **EnsurePlatformBufferSpaceFor**
- **FillUTF16Buffer**
- **PopulateUTF16Buffer**

Transform::TransformOrigin

TransformOrigin: The first argument is now const. This should never have been changeable. If your code changes this CoordinateSpace, it is in error.

metadata::ResourceRef

ResourceRef: A new optional argument is available as the third argument in this method. If you use none/one/two parameters, then it does not impact your code. Otherwise, if you use more than two parameters, insert origDocumentID as the third argument. This argument means the common identifier for the original document.