What's New in Adobe InDesign CS3 Products SDK

The Adobe® InDesign® CS3 Products SDK is a set of tools that helps you develop software that interacts with InDesign CS3 and InDesign CS3 Server technology. The SDK contains header files, type libraries, simple utilities, sample code, and documentation.

For the benefit of previous users, this document summarizes the changes to the SDK in this release.

C++ IDE Requirements

On Mac OS®, XCode 2.4.1 is now required for InDesign CS3 C++ plug-in development. CodeWarrior is no longer supported.

On Windows®, the required C++ development environment is now Visual Studio 2005.

Xcode Project Template

An InDesign SDK Project XCode template was created to ease the creation of InDesign plug-ins that are structured like an SDK sample. To learn more about the XCode template, see the "Creating an InDesign SDK Xcode Project" section in *Adobe InDesign CS3 Porting Guide* (`docs/guides/porting-guide.pdf`).

Porting Content

Porting content was developed to help you transition to InDesign CS3. *Adobe InDesign CS3 Porting Guide* discusses the main changes to the InDesign API, porting procedures, and how to transition to Visual Studio 2005 and XCode 2.4.1. The major changes to the InDesign CS3 API discussed in the document are as follows:

- **Command and Observer Changes** — The re-architecture of commands and observers is one of the most significant changes in InDesign CS3. The architecture for supporting undoable operations was revised. A command no longer must provide Undo and Redo methods that revert or restore changes made by its Do method. Instead, modifications to the database are monitored and recorded automatically in the database's command history. The application reverts the state of affected objects on undo and restores the changed state on redo. Also, a new kind of notification mechanism—lazy notification—was introduced to improve performance. The command and observer re-architecture provides better performance, simpler code, and greater stability.
- **PMString Changes** — PMString changed greatly in InDesign CS3. The changes include deprecation of platform methods and reinforcement of the restriction to use PMString only for user-interface strings.
- **Chooser Classes** — The chooser classes SDKFileOpenChooser, SDKFileSaveChooser, and SDKFolderChooser are defined in `SDKFileHelper.h`. The contract between these chooser classes and their clients was redefined in InDesign CS3 with regard to PMString changes. In InDesign CS3, client code is explicitly required to pass translatable strings for dialog titles and file-filter names to methods in these classes.

- **Overlap Manager Rewrite** — Since InDesign 1.0, text-wrap features have been managed by a set of interfaces and implementations collectively known as the spread-overlap manager. The spread-overlap manager was responsible for keeping track of which page items overlapped and for damaging text regions when necessary. To improve performance and support new feature requests, overlap management was rewritten in InDesign CS3. The ISpreadOverlapManager and ISpreadComposer interfaces were removed from kDocBoss, and ISpreadOverlapManager was removed from kSpreadBoss.
- **ITextFrame Replaced** — In versions of the applications prior to InDesign CS3, both kMultiColumnItemBoss and kFrameItemBoss aggregated ITextFrame(IID_ITEXTFRAME). This caused confusion and required coding conventions and utilities when dealing with an ITextFrame interface. This confusion was eliminated by replacing the ITextFrame interface on both bosses. Each boss has its own interface, and ID. kMultiColumnItemBoss aggregates IMultiColumnText-Frame (IID_IMULTICOLUMNTEXTFRAME), and kFrameItemBoss aggregates ITextFrame-Column (IID_ITEXTFRAMECOLUMN). `ITextFrame.h` was deleted, and IID_ITEXTFRAME is no longer defined.
- **Modernizing with Regard to System Types** — On Mac OS, the graphical user interface in InDesign CS3 and earlier versions is drawn using the old Control Manager and QuickDraw software. Some changes were made to support future efforts to move the applications to the more modern HIView and Quartz software.
- **Text Style Changes** — Beginning with InDesign CS3, users can group related styles into collections, called *style groups*. InDesign CS3 introduces the ability to create, edit, and delete folders in the lists in the Character Styles and Paragraph Styles palettes, as well as a user interface for navigating through those folders. This lets users have multiple styles with one name in different style groups, so there is no longer a need to have verbose names for styles to make them unique.
- **Scripting Changes** — The changes to the scripting architecture include new event-element resource, new RequestContext parameter added for IScript Methods, IPageItemScriptUtils updates, TypeInfo::GetTypeAsString and IScriptManager::GetTypeAsString signature changes, automatic scripting support for Units of Measure, IINXInfo::IsExcludedChild signature change, ScriptObject class changes, and Panels now being scriptable.
- **Palette and Panel-related APIs Changes** — Adobe adopted a new common palette appearance for all Adobe Creative Suite® 3 applications. Implementing this new appearance in InDesign required an extensive rewrite of InDesign's palette system, including changes to public APIs and how palette resources are specified.
- **Asset Management APIs Changes** — InDesign/InCopy® CS3 introduce a new Asset Management System Provider (AMSP) architecture for supporting asset management within InDesign and InCopy. The AMSP architecture enables support for Adobe Version Cue®, LiveEdit, and third-party asset-management system integration.
- **Transform and Geometry API Changes** — The transform APIs control the appearance of page items using transformation matrices, a 2D graphics technique. While this is well known mathematically, end users struggle with these features. In the interest of improving this functional area, the transform features were rewritten for InDesign Creative Suite 3, including a significant rewrite of the transform APIs.

Public Code Reorganization

There is a new folder, `source/public/components`, that contains source code (and utilities) for some actual InDesign components such as WidgetBin and Public. The folder is organized into the following subfolders:

- `publiclib` — Contains published source code from the Public library.
- `server` — Contains components pertaining to InDesign Server, such as CORBA regeneration and C++ SOAP client.
- `widget bin` — Contains public source code from the WidgetBin library.
- `xhtmlexport` — Contains code for the scripting-based XHTML Export Feature.

Scripting Support

Scripting Content

Previously available as a separate download, scripting content is now available as part of both the InDesign Products SDK and InDesign Server SDK. All scripting content is in the SDK's `scripting` folder, which contains scripting documentation and supporting scripts.

The InDesign Products SDK contains the following scripting content:

- *Adobe InDesign CS3 Scripting Tutorial*
- *Adobe InDesign CS3 Scripting Guide*
- *Adobe InDesign CS3 Server Scripting Guide*
- *Adobe InCopy CS3 Scripting Guide*
- JavaScript, AppleScript (MacOS only), and VBScript (Windows only) scripts for InCopy, InDesign, and InDesign Server.

The InDesign Server SDK contains the following scripting content:

- *Adobe InDesign CS3 Scripting Tutorial*
- *Adobe InDesign CS3 Scripting Guide*
- *Adobe InDesign CS3 Server Scripting Guide*
- JavaScript, AppleScript (MacOS only), and VBScript (Windows only) scripts for InDesign and InDesign Server

Scripting-based Feature Development

It is now possible to implement sophisticated features (like XHTML Export) entirely in scripting. To learn more, see *Feature Development With Scripting* (`docs/guides/feature-development-with-scripting.pdf`).

Scripting-based XHTML Export Feature

New sample code and documentation was written to demonstrate the scripting-based XHTML export feature. The source code is in `source/public/components/xhtmlexport/`. Documentation for the feature is in the "Building Blocks for using ExtendScript to Implement a Scripting Plug-in" section of *Feature Development With Scripting* (`docs/guides/feature-development-with-scripting.pdf`).

Improved DollyXs

The plug-in generation tool, DollyXs, was improved with the addition of selection suite and facade generation. To learn more about DollyXs, see the DollyXs Readme file (`devtools/sdktools/dollyxs/Readme.txt`).

Commands and Notification

The biggest architectural change to the InDesign CS3 API is the re-architecting of the Commands and Observers architecture to provide Auto Undo. These changes are discussed in the "Commands" and "Notification" chapters of *Adobe InDesign CS3 Products Programming Guide* (`docs/guides/programming-guide.pdf`) and in the "Command and Observer Re-architecture" section of *Adobe InDesign CS3 Porting Guide* (`docs/guides/porting-guide.pdf`). All SDK sample plug-ins also were rewritten to comply with the changes to the API.

Find/Change/Grep

New sample code and documentation was written to demonstrate the find/change/grep feature. The code is in: `source/sdksamples/codesnippets/SnpFindAndReplace.cpp/`. Documentation for the feature is in the "Find/Change Text" section of *Working With Text* (`docs/guides/ww-text.pdf`) and the Snippets > SnpFindAndReplace section of the API Reference (`docs/references/index.chm` or `docs/references/sdkdocs.tar.gz`).

Linguistics

The CHLinguistic SDK sample now demonstrates how to add support for ICU locale-based languages. Specifically, it demonstrates adding four languages for which there are no hard-coded LanguageIDs in `LanguageID.h`. Also, it was enhanced to include support for custom user dictionaries and a thesaurus service (available in InCopy) with six new languages. The CHLinguistic source code is in `source/sdksamples/chlinguistic`. Documentation for the source code is in the Samples > Text Samples > CHLinguistic section of the API Reference.

Transparency Effects

New sample code and documentation was written to demonstrate the improvements to the transparency effect feature. The code is in two directories: `source/sdksamples/transparencyeffect/` and `source/sdksamples/transparencyeffectui/`. Documentation for the transparency feature is in the "Transparency Effects" section of the "Graphics Fundamentals" chapter of *Adobe InDesign CS3 Products Programming Guide* (`docs/guides/programming-guide.pdf`).

Placing InDesign Files

InDesign CS3 provides the ability to place an InDesign file. Sample code demonstrating the Place feature is in `source/sdksamples/codesnippets/SnpPlaceFile.cpp`.

Table and Cell Styles

New sample code and documentation was written to demonstrate the table- and cell-styles feature. The code is in `source/sdksamples/codesnippets/SnpManipulateTableStyle.cpp`. Documentation for the feature is in the "Table and Cell Styles" section of *Working With Tables* (`docs/guides/ww-tables.pdf`).

SDK Workspace Extensions

Beginning in InDesign CS3, panels cannot be organized by specifying a palette ID in a Panel-List resource. By default, all third-party panels are grouped in one panel container, and the container is not visible at start-up, when there is no saved user data (preferences). As a result, when InDesign is installed and run for the first time, third-party panel(s) are not visible. When the user chooses to show the panel, it is shown in the same panel container as other third-party panels.

To allow third parties to group and show their panels when InDesign is launched for the first time, the workspace extension is introduced in InDesign CS3. Essentially, a workspace extension is a mini-workspace XML file located in a predetermined area, so InDesign reads the XML file at start-up and positions panels as specified in the workspace extension. For details about workspace extensions, see the "Organizing Panels with Workspace Extensions" section in *Working With User Interfaces* (`docs/guides/ww-user-interface.pdf`).

No SDK sample plug-in panels are included in the default InDesign workspace extension, so when you launch InDesign for the first time with SDK panels loaded, the SDK panels are not visible. To make the SDK panels visible at first launch, you can use one of the supplied SDK workspace extensions located at presets/InDesign Workspaces/workspace extensions/. To learn how to use the SDK workspace extensions, read the "SDK Workspace Extensions" section in *Working With User Interfaces*.

Quick Apply Dialog

New sample code and documentation was written to demonstrate the customization of the Quick Apply Dialog feature. The sample code is in the SnippetRunner plug-in code in `source/sdksamples/snippetrunner/SnipRunQuickApplyService.cpp`.

Documentation on the Quick Apply Dialog is in the "The Quick Apply Dialog" section of *Working With User Interfaces* (`docs/guides/ww-user-inerface.pdf`).

InDesign CS3 Server Java/CORBA Support

InDesign CS3 Server provides support for accessing your plug-in's functionality through Java/CORBA, through a static CORBA support plug-in and a Java JAR file. To add your plug-in's functionality to the CORBA support plug-in and Java JAR file, you need to regenerate these components. The InDesign CS3 Products SDK contains instructions for regenerating the InDesign Server Java/CORBA support files. see *Regenerating the Adobe InDesign CS3 Server Java API* (`docs/guides/regenerating-java-api.pdf`).

InDesign CS3 Server SDK

We are introducing a new SDK for InDesign CS3 Server. This SDK contains documentation and sample code specific to InDesign Server and can be used in conjunction with the shipping CORBA Support plug-in, to interface with InDesign Server using Java. If your system requires additional plug-ins, you must regenerate CORBA support using the InDesign CS3 Products SDK, as described above.

The InDesign CS3 Server SDK contains the following:

- Documentation including *Introduction to Adobe InDesign CS3 Server*, *Regenerating the Adobe InDesign CS3 Server Java API*, *Working With Adobe InDesign CS3 Server Java,* and the InDesign Server Java API Reference.
- `InDesignServerAPI.jar`, a Java JAR file containing the entire InDesign Server Java API.
- Sample Java components:
  - `sampleclient-java-corba` — A sample client application that runs a JavaScript script under InDesign Server via CORBA.
  - `sampleclient-java-soap` — A sample client application that runs a JavaScript script under InDesign Server via SOAP.
  - `snippets` — The Java equivalent of the scripts in *Adobe InDesign CS3 Scripting Guide*.
- Scripting documentation and scripts, as discussed in "Scripting Content" on page 3.