Adobe

Adobe Acrobat DC SDK Samples Guide

2/3/21 Adobe, Inc.

Contents

1 Plugin Samples

BasicPlugin

Location

Plugins/Samples/BasicPlugin

Description

Provides the minimum code required to enable developers to get started. The code adds a new menu item under the Acrobat SDK menu and displays a simple message.

With the basic plugin framework set up in the sample, users can quickly make a plugin of their own by modifying and adding the code in the BasicPlugin.cpp file to set up their own menu item and menu function.

BatesNumbering

Location

Plugins/Samples/BatesNumbering

Description

Demonstrates Bates Numbering features exposed to third-party developers:

- How to extract Bates Numbering that is added by Acrobat.
- How to use Acrobat API to programmatically add Bates Numbering to PDF documents.
- How to use Acrobat API to programmatically remove Bates Numbering from PDF documents.

Usage

Follow these steps to test the functionality of the sample:

Open a PDF document.

- Click Acrobat SDK > Bates Numbering > Add... to add Bates Numbering to the PDF document.
- Click Acroabt SDK > Bates Numbering > Extract... to extract Bates Numbering from the PDF document.
- Click Acrobat SDK > Bates Numbering > Remove... to remove Bates Numbering from the PDF document.

Note: This plugin will not work in Adobe Reader.

CapiSamples

Location

Plugins/Samples/CapiSamples

Description

There are two samples in this demonstration. One is SampleCsp. It is a DLL, and it is the actual Cryptographic Service Provider (CSP). The SampleRegistrar sample uses the services provided by SampleCsp.

The purpose of SampleCsp is to provide a reference implementation. The cryptographic functionality of this CSP is passed through to the Microsoft Enhanced Crypto Provider. This implementation currently works on PFX files. SampleCsp opens the PFX files and uses the credential within them.

SampleRegistrar is a separate application which allows users to manage the certificates bound to SampleCsp. This application interacts with SampleCsp to do the following:

- Register a digital ID file (PFX file) and add its certificates to the windows "My" store.
- Un-register a digital ID file and remove its certificates from the windows "My" store.
- List the registered digital IDs.

Usage

This sample requires a CSP digitally signed by Microsoft. Download Microsoft's CSDK from https://www.microsoft.com/en-us/download/details.aspx?id=30688.

Once signed, locate SampleCsp in the MsSignedSampleCsp directory.

To manage certificates by SampleRegistrar:

1. Compile SampleRegistrar.

2. Register SampleCsp by doing the following:
   - At Dos prompt, go to the MsSignedSampleCsp directory.
   - Type in Command 'regsvr32 SampleCsp.dll'.

3. At the DOS prompt, go to the directory where SampleRegistrar.exe resides.

4. Use the following commands to manage certificates:
   - To list registered PFXs: 'SampleRegistrar.exe -l'
   - To register PFX: 'SampleRegistrar.exe -r mycert.pfx password'
   - To un-register PFX: 'SampleRegistrar.exe -u <hex-encoded-sha1-cert-digest>'

There are some easy tests you can run to check the CSP is working with Acrobat:

1. Register a digital ID via the CSP. It must show up in the Security Settings under "Windows Digital IDs".

2. Try to make a digital signature with the registered digital ID.

3. Encrypt a file for this certificate and then try to decrypt it. You need to close the document and reopen it.

If all of these tests pass, then the CSP is working. The effect should be to un-register the digital ID, not to actually delete the file.

Note: This plugin is Windows only.

DdeServer

Location

PluginSupport/Samples/DdeServer

Description

Demonstrates how to establish communication between an external application and an Acrobat plugin using DDE. This allows a plugin to expose functionality to external applications that are not present in the standard IAC interfaces.

Usage

A sample client application is provided with the DdeServer plugin. The application communicates with the plugin to obtain the page number associated with the active page view and to add a text annotation to the page associated with the active page view. Acrobat must be running for the client application to function correctly.

Limitations

- Windows only.
- This plugin will not work in Adobe Reader since it accesses annotations. It is possible to use this technique to communicate with an Adobe Reader plugin.
- Non functional when Protected Mode is enabled.

DMSIntegration

Location

Plugins/Samples/DMSIntegration

Description

Demonstrates how a DMS system can integrate with Acrobat. This sample demonstrates the following three steps needed for DMS systems to integrate with Acrobat:

1. Write an ASFileSys for the DMS system.

2. Replace the Acrobat dialogs.
   - AVAppChooseFolderDialog

- AVAppOpenDialog
- AVAppSaveDialog

DocSign

Location

Plugins/Samples/DocSign

Description

Demonstrates how to use the PubSec API. It is more a "skeleton" plugin than a fully functional sample, since it has no encryption library. The plugin assumes a plugin author knows how to generate public/private key pairs, store them, use them to encrypt data, etc. This skeleton sample demonstrates how to hook into Acrobat. plugin does.

Usage

To use DocSign, change the default signature handler in Acrobat Preferences to the DocSign handler. After that, DocSign will be used through the same user interface points as the standard signature handler.

Implementation details

This sample does not provide true digital signatures. It provides a skeleton of a PubSec API-based plugin. Developers must add your own encryption routines.

Embed3DData

Location

PluginSupport/Samples/Embed3DData

Description

Demonstrates how to programmatically create a 3D annotation in a PDF file. For more information about the process, see the chapter titled "Creating 3D Annotations" in the Developing Plugins and Applications Guide (plugin_apps_developer_guide.pdf)

The program output is a new PDF file with the 3D annotation.

Implementation details

- Create a new PDF file with one page.
- Add a 3D annotation to the page.
- Create the 3D stream with the input 3D data.
- Create entries for the 3D stream dictionary.

- Embed optional JavaScript code for the 3D stream.
- Create optional "VA" entry for the 3D stream.
- Create optional animation dictionary for the 3D stream.
- Specify key-value pairs in the annotation dictionary.
- Create optional activation entry for the 3D annotation.
- Specify option default initial view entry for the 3D annotation.
- Create an annotation appearance for the 3D annotation. The code shows two possible ways to create a form XObject for the 3D annotation's appearance. The annotation appearance can come from either a PDF file containing an image or a watermark generated on the fly.
- The output PDF file works with Acrobat and Adobe Reader 7.0 or later.

Usage

- Select "Create PDF with 3D Annotation" from "Advanced-Acrobat SDK".
- Select the U3D model file when prompted.
- Choose whether you want to specify a poster for the 3D annotation when prompted.
- Select the poster PDF file that has the image if you answered "Yes" in step 3.

Note: This plugin does not work in Adobe Reader.

RplcFileSystem

Location

Plugins/Samples/RplcFileSystem

Description

Demonstrates how to create a custom file system. The sample is basically a wrapper around the default file system. The point is to demonstrate the mechanism for adding a new file system without becoming bogged down with complex implementation considerations.

Usage

The replacement file system can be exercised by selecting the Replacement File System menu item on the Acrobat SDK submenu.

Implementation details

The sample only implements the callbacks supported by the default file system.

SampleExtn

Location

Plugins/Samples/SampleExtn

Description

This sample demonstrates how a plugin can work with the app in sandbox mode (protected mode is enabled). For details on how the plugin and broker interact with each other and hence the app, please refer 'Sandbox Broker Extensibility' the Acrobat SDK Overview.

Usage

1. Configure the sample to enable it for the app and build the plugin.
2. Copy the SamplePI.api to <Installed_Path>\plug_ins\SamplePI.api.\
3. Copy the SamplePIBroker.exe process to <Install_Path>\plug_ins\pi_brokers\SamplePIBroker.exe.
4. Start the app in Protected Mode and play with SamplePI.api from the Acrobat SDK->SamplePI menu.

Note: This is a Windows-only plugin

SelectionServer

Location

PluginSupport/Samples/SelectionServer

Description

Demonstrates how to implement a minimal selection server. The SelectionServer plugin implements a selection server to handle images. The image selection tool allows users to select images by calling AVDocSetSelection() with a selection type of "Image". The selection server functionality is limited to getting, showing, and losing a selection.

Usage

The plugin registers an Image Selection tool that allows users to select image XObjects on the page.

Installs the Acrobat SDK > Image Selection Tool menuitem and the associated AVToolButton. You can use either to toggle the state of the selection server.

Note: Will display various messages using the SnippetRunner plugin HFT in the CommonInterface.air. You must compile and run the SnippetRunner sample and CommonInterface.air prior to running this sample.

Note: Will not work in Adobe Reader.

Snippet Runner

Location

PluginSupport/Samples/SnippetRunner

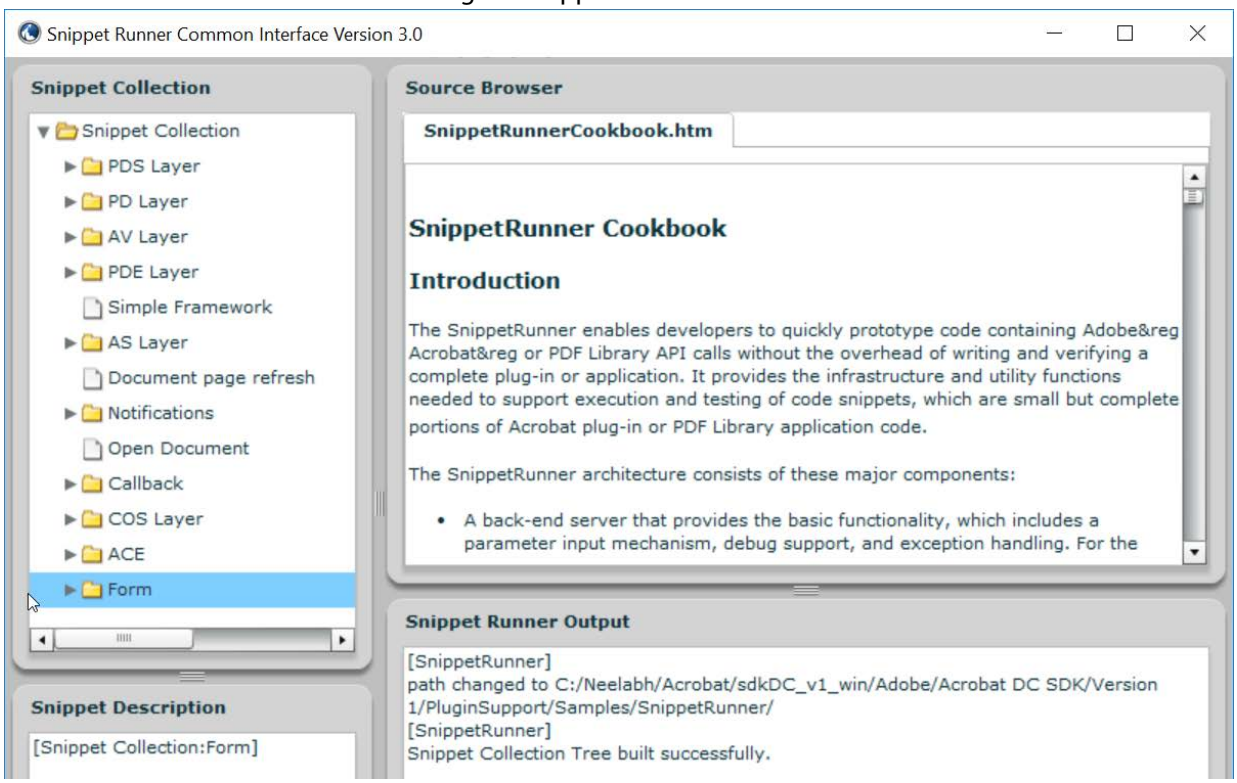
Description

The SnippetRunner plugin provides infrastructure and utility functions to support execution of Acrobat plugin code snippets. A snippet is a small complete portion of Acrobat plugin code. More than 100 sample code snippets are provided in the SnippetRunner project to demonstrate Acrobat API methods. However, the SnippetRunner plugin also allows developers to quickly prototype Acrobat API calls without the overhead of writing and verifying a complete plugin.

While this document includes information on the SnippetRunner environment and use, for more details on writing and loading original snippets, see the SnippetRunner Cookbook.

Implementation details

SnippetRunner interaction is via a common AIR-based graphical user interface for Windows and Mac. This Common User Interface acts as a client to its associated SnippetRunner back-end -- the SnippetRunnerServer Acrobat plugin -- and provides Acrobat with a common cross-platform GUI. By sending commands to the SnippetRunnerServer back-end and receiving feedback from it, the Common Interface executes snippet commands and provides rich information about snippets, output and document status. It also allows browsing the snippet source code from within the interface.



Usage

Some snippets require external files. These can either be sample files for input or resources (for UI artifacts). Example files are delivered within the `Examples` directory of the SnippetRunner plugin.

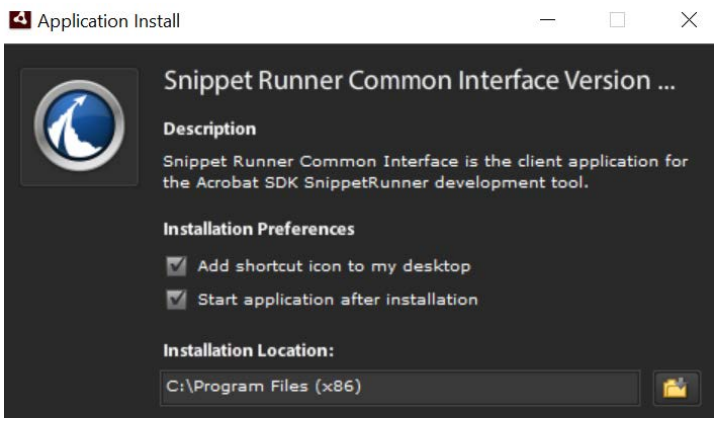
Note: This plugin will not work in Adobe Reader.

When starting Common Interface for the first time, set the root of the `SnippetRunner` folder. This persists between application instantiations. Note that for security, the root SnippetRunner folder must reside in the same partition as the boot partition.

When there's a PDF document open in a browser prior to explicitly start Acrobat, subsequent connection attempts from the Common Interface will be made with the SnippetRunnerServer plugin associated with the Acrobat browser plugin process rather than the Acrobat process you just started. If your Common Interface suddenly fails to connect with the SnippetRunnerServer or the changes you made to a snippet are not reflected after rebuilding the plugin and restarting Acrobat, quit your browser and restart Acrobat and the Common Interface.

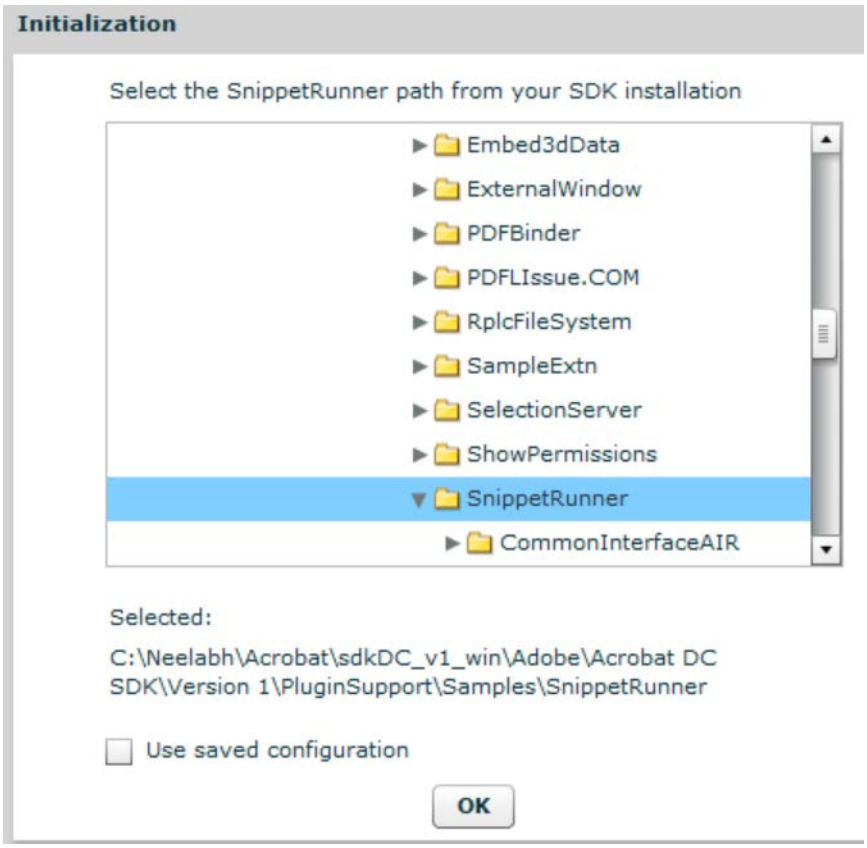
To run the SnippetRunner:

1. Build the SnippetRunner sample.
2. Copy the SnippetRunnerServer.api to C:\Program Files (x86)\Adobe\(product version)\Acrobat\plug_ins.
3. Install last available working version of Adobe AIR runtime from: https://get.adobe.com/air/. Doing so installs AIR to C:\Program Files (x86)\Common Files\Adobe AIR\Versions\1.0.
4. Add the AIR path above to your machine's PATH environment variables.
5. Open the command prompt as an Administrator.
6. Set the Snippet Runner directory. For example: cd C:\(username)\Acrobat\sdkDC_v1_win\Adobe\Acrobat DC SDK\Version 1\PluginSupport\Samples\SnippetRunner
7. Run the command: "Adobe AIR Application Installer.exe" -ignoreExpiredCertificateTimestamp
8. Navigate to CommonInterfaceAIR.air in the Snippet Runner folder inside Acrobat SDK and click install.
9. Change the install location to the Snippet Runner directory where CommonInterfaceAIR.air is present.



10. Complete the installation

11. Once the CommonInterfaceAIR.air launches, select the Snippet Runner Folder path as the initial path. Doing so should load the snippet list.



12. Launch Acrobat so that it will act as server for SnippetRunner's CommonInterfaceAIR.air client.

Included snippets

AVAlertSnip

AVAppFrontDocChangeNotSnip

AVAppPrefsSnip

AVAppRegisterForContextMenuSnip

AVAppRegisterForPageViewDrawingSnip

AVDocAVWindowDidChangeSnip

AVDocCloseNotSnip

AVDocGetSelectionTypeSnip

AVDocOpenNotSnip

AVDocShowAnnotPropertiesSnip

AVDocWindowWasAddedSnip

AVDocWindowWasRemovedSnip

AVEnumActionHandlerSnip

AVPageViewDrawRectSnip

AVPageViewToggleWireframeDrawingSnip

AVPrintSnip

AVSaveAsRtfSnip

AVWindowMaximizeCurrentPageViewSnip

CallJsCreateButtonSnip

CallJsGetFieldValueSnip

CallJsResponseSnip

CallJsSignatureFieldSnip

CallJsTextFieldSnip

CloseFrontDocSnip

ColorSelectedBookmarksSnip

ColorSetupSnip

ConvertFlate2JPXSnip

CosDecryptDataSnip

CosEncryptDataSnip

DocMetadataSnip

EmitPostScriptSnip

EnumAVConvExtSnip

ExportFormDataSnip

FormCalculationsSnip

IdleProcSnip

InvokeAccessibilityCheckerCmdSnip

InvokeCreateAllThumbsCmdSnip

InvokeDeleteAllThumbsCmdSnip

InvokeDeleteCmdSnip

InvokeFlattenOCGsCmdSnip

InvokeMakeAccessibleCmdSnip

InvokeOpenOptionsCmdSnip

InvokeSummarizeCmdSnip

LoadHowToSnip

OpenDocumentSnip

OptContNotificationTracerSnip

PDEPathToggleVisibilitySnip

PDPageNotifyContentsChangeSnip

RegisterFileConverterSnip

ReplaceMethodSnip

ResetFormSnip

SeparationsPreviewSnip

ShowTextFieldNamesSnip

SmartPDPageSnip

SnapZoomSnip

TextInfoSnip

TransformMetadataSnip

TransHandlerSnip

ACEEnumProfilesSnip

ACEEnumSettingsSnip

ACEGetWorkingSpaceSnip

ACETransPDETextColorSnip

AddGlyphsSnip

AddImageMetadataSnip

AddImageSnip

AddPageMetadataSnip

AddStructureSnip

AddTagSnip

AddXObjectStructureSnip

ASBigFileSnip

ASCabPutGetSnip

ASChangeTempFileSysSnip

ASDateSnip

ASFileIteratorSnip

ASGetConfigurationSnip

ClassMapSnip

ConvertOCGsToRadioButSnip

CosCryptGetVersionSnip

CosDictKeyNameStringSnip

CosDoc64Snip

CosNumberObjRangeSnip

CosObjCompressionSnip

CosObjDecompressionSnip

CosObjectExplorerSnip

CosObjWeakReferenceSnip

CosStream64Snip

CreateAnnotOCSnip

CreateContentXORSnip

CreateDocStructSnip

CreateImageContentOCSnip

CreateTextContentOCSnip

ExploreMetadataSnip

ExploreStructSnip

FontInfoSnip

GetDocKeywordSnip

GetDocMetadataSnip

ImageInfoSnip

JPXColorSpaceExplorerSnip

JPXPaletteExplorerSnip

MakeBookmarkSnip

ObjShiftSnip

OCActionControlSnip

OCGUIReorderSnip

OCTextAutoStateSnip

PDCreateMasterOCGSnip

PDDocDidDeletePagesNotSnip

PDEContentExplorerSnip

PDEPathDrawCurveSnip

PDEPathDrawLineSnip

PDEPathDrawRectSnip

PDEPathExplorerSnip

PDOCConfigCreateSnip

PDOCConfigExplorerSnip

PDOCGChangeLockedStateSnip

PDOCGExplorerSnip

PDOCGToggleIntentSnip

PDOCSetDefaultConfigSnip

PDPageSetTransparencySnip

RaiseExcepSnip

RemoveEmbeddedFontSnip

RoleMapSnip

SecureDocumentSnip

SetDocBaseURLSnip

SimpleSnip

TextChangeColourSnip

TextExtractionSnip

UserPropertiesExplorerSnip

Stamper

Location

PluginSupport/Samples/Stamper

Description

Demonstrates how to implement an AVTool and an annotation handler. It also demonstrates use of the AVUndo API. It is accessed from the Acrobat SDK submenus' Stamper Annotations menu item.

Usage

Stamper annotations can be added to documents by selecting the Stamper tool and dragging a rectangle that defines the desired boundary for the annotation.

Note: Will not work in Adobe Reader.

Starter

Location

PluginSupport/Samples/Starter

Description

A plugin template that provides a minimal implementation for a plugin. Developers may use this plugin as a basis for their own plugins.

UncompressPDF

Location

Plugins/Samples/UncompressPDF

Description

A utility that removes all compression from the page content and form XObject streams within a PDF document. The sample illustrates how to implement an AVConversionFromPDFHandler, an object that is used to allow users to save PDF documents in a format other than those supported by Acrobat.

Usage

The functionality of the sample is accessible through a file type filter in the File > Save As dialog box. The filter name is "Uncompressed PDF Files". When any document is saved as that file type, the compression is removed from all of the document's page content and form XObject streams.

Implementation details

The sample uses the PDF Consultant framework to locate all of the page content and form XObject streams in the document.

Note: Will not work in Adobe Reader.

WeblinkDemo

Location

PluginSupport/Samples/WeblinkDemo

Description

Demonstrates how to register a new Weblink driver with the Weblink plugin. This could be used to communicate with a web browser or application that is not included in the standard list, or to send information to the Netscape browser through a different interface than the standard Weblink driver.

Usage

The sample simply highlights the interaction between Acrobat and a Weblink driver. Each callback writes a message to the Common Interface window indicating that it has been called.

Note: This sample requires the CommonInterface.air window in the SnippetRunner plugin. The SnippetRunner sample must be compiled and running prior to running the WebLinkDemo sample.

wxPlugin

Location

Plugins/Samples/wxPlugin

Note: This sample requires the third-party, open-source, cross-platform wxWidgets headers and libraries. The headers and libraries must be downloaded from www.wxwidgets.org prior to creating or compiling a project. This is one open-source, cross-platform UI technology Adobe has tried with Acrobat developer plugins. Feel free to investigate alternatives if you would rather not use wxWidgets.

Description

Provides a sample on how to integrate a third-party UI framework.

Usage

To integrate wxWidgets into your plugin, follow the steps below for both the platform specific section, and the 'Both' section.

Both platform projects assume there is a NonAdobeSupport directory at the same level as PluginSupport. Inside this would be a wxWidgets/mac/ or wxWidgets/win/ directory containing the sub-directories 'include' and 'libraries', with the include directory having the contents of wxWidgets/include/, and the libraries directory containing the contents of the lib directory, including the lib/wx/ directory.

To create a project using wxWidgets:

1. Define WIN32_LEAN_AND_MEAN, __WXMSW__ and wxUSE_GUI in the project settings
2. Add header paths for ..\..\..\..\NonAdobeSupport\wxWidgets\win\include;..\..\..\..\NonAdobeSupport\wxWidgets\win\libraries\mswd\;..\..\..\..\NonAdobeSupport\wxWidgets\win\include\msvc\ to C/C++->Preprocessor->Preprocessor Definitions.
3. Add ..\..\..\..\NonAdobeSupport\wxWidgets\win\libraries to Linker->General->Additional Library Directories
4. Add wx libraries to Linker->Input->Additional Dependences. You probably need at least wxbase3xd.lib and wxmsw3xd_core.lib.
5. This plugin requires wxWidgets 3x. Compile wxwidgets and copy the generated libs to NonAdobeSupport\wxWidgets\win\lib and header files to NonAdobeSupport\wxWidgets\win\include. This path may differ as per project configuration.
6. Add Headers/SDK/wxInit.cpp to your project to get the PluginApp and AcrobatFrame classes.
7. Add wx initialization and tear down routines to your 'Plugin'Init file. See the sample for details.
8. Include necessary headers. Use the form wx/foo.h for namespace clarity.
9. Create a dialog class based on wxFrame or wxDialog.

CustomTool

Location

Plugins/Samples/CustomTool

Description

Demonstrates method to add a custom tool in the tool center with scalable customizable pdf icons. The plugin adds two sub-tools in the top-bar. These sub-tools also have the scalable pdf icons. The first sub-tool lists all the documents currently opened in the Acrobat and demonstrates the usage of the AVDocGetDisplayTitle API. The second sub-tool brings the next document to front and demonstrates the usage of AVDocBringToFront API.

Usage

Follow these steps to test the functionality of the sample:

1. Open more than one PDF document. If multi-tab functionality is turned on, each document opens in a separate tab, else each document opens in a separate window.
2. Open Tools > Add-ons > Multitab Utilities. A toolbar with two tools "Switch to next Doc" and "List all document titles" appears.
3. Click "Switch to next Doc" to move to the next document.
4. Click "List all document titles" to open a message box containing titles of all the open documents.

2 JavaScript Samples

JavaScript Samples Portfolio

Location

JavaScript/Samples/Portfolio

Description

A portfolio (a new feature introduced in Acrobat 9.0) which packages and organizes all Acrobat SDK JavaScript samples into folder hierarchies.

Upon opening this portfolio document, the embedded document JavaScript enumerates the content of the portfolio and prompts the user information about the details of files contained in the portfolio. The "Install Folder-level JavaScript Samples" button facilitates installation of folder-level JavaScript samples contained in the portfolio to your user `JavaScript` folder.

Notice that on Windows Vista, the user JavaScript folder is hidden by default. Please modify your file system folder option to reveal hidden folders and files prior to initiating the folder-level JavaScript installation to ensure uninterrupted installation process.

AddSignature

Location

JavaScriptSupport/Acrobat SDK JavaScript Samples Portfolio.pdf/Home/JavaScript Samples/Non-Embedded JavaScript/

Description

Shows how to programmatically sign a PDF document using a predefined digital ID file. The JavaScript code includes all the digital signature information used to sign the document, except the path and password for the digital ID file.

When you open the Acrobat SDK JavaScript Samples Portfolio.pdf, a dialog prompts you with basic information about sample files contained in the portfolio. Clicking on the "Install Folder-level JavaScript Samples" button and follow the steps to copy JavaScript samples to your user's `JavaScript` folder. When you restart Acrobat you should see a new item Add My Signature under the Edit/Acrobat SDK JavaScript menu.

When you are ready to sign a PDF document, click the newly added Add My Signature menu item. After you input the platform-independent path and the password through a dialog box, the program creates a digital signature field in the top left corner. The path and password are valid in an Acrobat session, so you can continue to sign more documents in the session without the input dialog box. If you change the

JavaScript code to specify the path and password for the digital ID file to be used, then when you click the menu item, the program will automatically sign PDF documents without requiring the UI. A digital signature file (DrTest.pfx) is provided with the sample. To use it, put it in a folder, and specify the proper DIpath (for example `/C/DrTest.pfx`). The password is "testpassword".

It is also possible to execute the JavaScript code to sign a PDF document from another JavaScript program, or from a plugin, Visual Basic (using interapplication communication), or Visual C++ program through the function ExecuteThisScript. See the source code for more information.

To execute the security-restricted method through a menu event in this sample, the item labeled Enable Menu Items JavaScript Execution Privileges under Edit > Preferences > JavaScript must be checked. An alternative way is to wrap up the security methods used in the code through a trusted function. For details and examples, see app.trustedFunction in the *JavaScript for Acrobat API Reference*.

Note: Requires Reader-enabled PDF document to run in Adobe Reader.

AddToolbarButton

Location

JavaScriptSupport/Acrobat SDK JavaScript Samples Portfolio.pdf/Home/JavaScript Samples/Embedded JavaScript/

Description

This sample shows how to add/remove toolbar buttons in Acrobat using the Acrobat JavaScript APIs. The resulting button has warning text below it that tells the user that the added tool is a JavaScript window, as a security measure.

Usage

Click the Add Toolbar Button to add a toolbar button of a running man. When you click that item, you will see a message "You have clicked the JavaScript toolbar button". Click the Remove Toolbar Button to remove the toolbar button.

AnnotatedWords

Location

JavaScriptSupport/Acrobat SDK JavaScript Samples Portfolio.pdf/Home/JavaScript Samples/Non-Embedded JavaScript/

Description

This script returns to the console the words covered by highlight annotations. The script can be adapted to output the text elsewhere, such as to a file. It can also be edited to include other text-related annotations, such as underline or cross-out.

This script is limited to single-column, left-to-right, top-to-bottom, horizontal, non-overlapping text. Vertical text, text bound to a shape, right-to-left text, etc., may not work.

Annotated text is reported annotation by annotation in the order that the annotations were applied to the doc (appear in the annotations array), not in reading order or any other word order on the page.

This script uses doc.getPageNthWordQuads() to obtain its results. As such, it cannot report partial words under an annotation. Instead, any word that is partially covered by an annotation is included in the output. Additionally, line spacing, document origin, font size and other factors may affect how closely an annotation's quads and a word's quads overlap. For some documents, the results may not be accurate. Please see in-line comments for more information.

Usage

When you open the Acrobat SDK JavaScript Samples Portfolio.pdf, a dialog prompts you with basic information about sample files contained in the portfolio. Clicking on the "Install Folder-level JavaScript Samples" button and follow the steps to copy JavaScript samples to your user's `JavaScript` folder. When you restart Acrobat you should see a new item Copy Annotated Words under the Edit/Acrobat SDK JavaScript menu.

The script will work on the active document and output words covered by highlight annotations to the console in a page-by-page list.

Note: Requires Reader-enabled PDF document to run in Adobe Reader.

AnnotSample

Location

JavaScriptSupport/Acrobat SDK JavaScript Samples Portfolio.pdf/Home/JavaScript Samples/Non-Embedded JavaScript/

Description

Folder level JavaScript code to exercise the annotation APIs useful in reviewing workflow. It can be used with a rights-enabled PDF document in Adobe Reader as well as regular PDF documents in Acrobat.

Usage

When you open the Acrobat SDK JavaScript Samples Portfolio.pdf, a dialog prompts you with basic information about sample files contained in the portfolio. Clicking on the "Install Folder-level JavaScript Samples" button and follow the steps to copy JavaScript samples to your user's `JavaScript` folder. When you restart Acrobat you should see a new item Annotation Sample under the Edit/Acrobat SDK JavaScript menu.

It will trigger a JSADM dialog box to show the following functions:

- Set annotations as read only or editable
- Import annotations from a local FDF file
- Export all annotations to a local FDF file

- Export editable annotations to a local FDF file

You can try the functions while creating or modifying annotations in the PDF document.

To run the sample, you need to specify a path in the dialog box for a data repository in your environment. The path must be a safe path and in a platform-independent format, such as `/c/test/myAnnotDataFile.fdf`.

A test file, ReaderEnabledSample.pdf, with Reader-enabled usage rights is provided in the sample folder.

Note: Requires Reader-enabled PDF document to run in Adobe Reader.

CallMediaActionScript

Location

JavaScriptSupport/Acrobat SDK JavaScript Samples Portfolio.pdf/Home/JavaScript Samples/Multimedia/

Description

This sample demonstrates the ability to invoke ActionScript methods embedded in the Rich Multimedia Annotation from Acrobat JavaScript.

ActionScript methods to be exposed to the ExternalInterface of the container (Acrobat Flash Framework) register themselves as callable from the container via the ExternalInterface.addCallback method.

ConvertDate

Location

JavaScriptSupport/Acrobat SDK JavaScript Samples Portfolio.pdf/Home/JavaScript Samples/Embedded JavaScript/

Description

ConvertDate.pdf is a JavaScript sample that demonstrates how to convert the PDF date format to a JavaScript date object and back again. It also shows how to display the JavaScript date in various formats using utility methods.

Usage

There are three operation groups with instructions. Click the buttons and check results shown in text fields. You can convert the PDF date to a JavaScript date object with various formats, input your own date in the PDF format to check the conversion result, and convert the JavaScript date back to the PDF format.

DeleteNoCommentPages

Location

JavaScriptSupport/Acrobat SDK JavaScript Samples Portfolio.pdf/Home/JavaScript Samples/Non-Embedded JavaScript/

Description

DeleteNoCommentPages is a folder-level JavaScript code that can be useful in review workflows. It is similar to the Acrobat 6 Summarize Comments option which deleted pages without comments as it summarized.

Usage

When you open the Acrobat SDK JavaScript Samples Portfolio.pdf, a dialog prompts you with basic information about sample files contained in the portfolio. Clicking on the "Install Folder-level JavaScript Samples" button and follow the steps to copy JavaScript samples to your user's `JavaScript` folder. When you restart Acrobat you should see a new item Delete Pages without Comments under the Edit/Acrobat SDK JavaScript menu.

To run the sample, select Edit > Acrobat SDK JavaScript > Delete Pages Without Comments.

This script provides a status message (alert) upon completion. It can be removed for silent operation, such as batch processing. The processed document is not saved.

Note: Requires Reader-enabled PDF document to run in Adobe Reader.

EventState

Location

JavaScriptSupport/Acrobat SDK JavaScript Samples Portfolio.pdf/Home/JavaScript Samples/Multimedia/

Description

Demonstrates two ways for event listeners to have a local persistent state. It is particularly helpful to developers in writing multimedia event listeners. The boxes in the top row are ScreenAnnots with event listeners that log events to the list boxes below. Click each one to start logging, then move the mouse around among them and click some more to watch the events being logged.

The main functions are document-level JavaScript code. The two ScreenAnnots on the left use local variables and nested scope, and the two on the right use properties in the event listener object.

System Requirements. Acrobat 6.0 or later. The machine should be ready for multimedia play - sound card, speaker, proper system settings, and a multimedia player, such as Windows Media Player, QuickTime, or Flash.

GoToBookmark

Location

JavaScriptSupport/Acrobat SDK JavaScript Samples Portfolio.pdf/Home/JavaScript Samples/Non-Embedded JavaScript/

Description

Provides a utility for users to get to a bookmark in a PDF document in Acrobat. If found, it executes the bookmark action defined in the PDF file. Usually this results in a page view, but other bookmark actions are possible.

The function GoToBookmark, demonstrated in this sample file, could be used, for instance, in accessing Help information in PDF documents. A help function could call this JavaScript method, specifying the PDF Help document and the bookmark to be found. The PDF page referenced by that bookmark would then be shown if the search succeeds.

When you open the Acrobat SDK JavaScript Samples Portfolio.pdf, a dialog prompts you with basic information about sample files contained in the portfolio. Clicking on the "Install Folder-level JavaScript Samples" button and follow the steps to copy JavaScript samples to your user's `JavaScript` folder. When you restart Acrobat you should see a new item Go To Bookmark under the Edit/Acrobat SDK JavaScript menu. Click on it to get a dialog box to fill in your search criteria.

The input string is the full name of a bookmark to be found. The search is case insensitive. For example, if you open this Samples Guide PDF file, select the Go to Bookmark... menu item, enter the string "GoToBookmark", and click OK, you will go to the beginning of this section because that heading is the first bookmark with the given name.

You may also specify the hierarchy level of your search in various ways, as in these examples:

`SDKJSSnippets` - Gets the first match in any level

`"Guide to SDK Samples:JavaScript Samples:Inside PDF:SDKJSSnippets"` - A completely specified hierarchy in which each token is one level down from the previous.

`"Guide to SDK Samples:*:SDKJSSnippets"` - A wildcard hierarchy in which "*" means there may be any number of levels there, including no level between.

The search process may be time consuming for a PDF document with a large number of bookmarks. To cancel the process, press Esc on Windows or Command-period on Mac OS. A progress bar is implemented using the thermometer JavaScript object.

JSCollection

Location

JavaScriptSupport/Acrobat SDK JavaScript Samples Portfolio.pdf/Home/JavaScript Samples/

Description

A collection of JavaScript snippets organized by function and fully indexed for easy access. It includes sections for Acrobat forms(field manipulation, data validation, etc.) and for documents (bookmarks, navigation, etc.). This collection provides a range of basic JavaScript samples which can be cut-and-pasted into a PDF document to perform basic tasks or to help build larger workflow solutions.

Note: Requires Reader-enabled PDF document to run in Adobe Reader.

JSCollectionDemo

Location

JavaScriptSupport/Acrobat SDK JavaScript Samples Portfolio.pdf/Home/JavaScript Samples/Embedded JavaScript/

Description

Includes several JavaScript snippets that demonstrate Acrobat JavaScript objects and methods. Each page in this sample document contains an independent piece of JavaScript sample code. The availability of JavaScript objects depends on the viewer type, platform, and code location, so some snippets have certain restrictions. For example, Text-To-Speech is a Windows-only sample; and Use of Template, Add Links, and Metadata do not work on Adobe Reader.

Contents:

- Execute a JavaScript - assigns text input of JavaScript code to a button action and executes the input code with the button.
- Popup Menu - creates a popup menu.
- Metadata - shows a quick way to get document metadata in XML-formatted text.
- Full screen - displays a PDF document as an automatically-progressing, full-screen slideshow.
- Progress Bar - displays a progress bar at the bottom of the Acrobat viewer window.
- Add Links - creates new links at specified words in a page.
- Text-To-Speech - presents a simple sample to speak the user's input text.
- Calculator - presents a functional calculator made by Acrobat forms with JavaScript.
- Show/Hide Text - shows text blocks one-by-one at the touch of a hidden button, as in a PDF presentation.
- Conditional Text - shows or hides fields as might be used in a PDF presentation.
- Printing - controls document printing through JavaScript print parameters.
- Use of Template - adds a new page using a predefined page template.

Due to the functionality restriction, the following snippets won't work on Adobe Reader:

- Metadata
- Add Links
- Text-To-Speech
- Use of Template

There are annotations on each page in the file. Typically, the notes with the "Help" icon (question mark) provide help to users in running the snippet and the notes with the "Comment" icon (word balloon) are hints about how the snippet is implemented. Most of the JavaScript code is attached; users can look at it and modify it to experiment.

Note: Will not run in Adobe Reader.

OCGLayerControl

Location

JavaScriptSupport/Acrobat SDK JavaScript Samples Portfolio.pdf/Home/JavaScript Samples/Embedded JavaScript/

Description

Demonstrates JavaScript APIs for PDF optional content groups. The code is embedded as document level and form field JavaScript.

Usage

Exercise the sample by clicking the various buttons. The functions are self-explanatory.

Note: Will not run in Adobe Reader.

PresentationMonitor

Location

JavaScriptSupport/Acrobat SDK JavaScript Samples Portfolio.pdf/Home/JavaScript Samples/Non-Embedded JavaScript/

Description

Creates a set of tools to monitor the progress of a presentation using PDF slides.

When you open the Acrobat SDK JavaScript Samples Portfolio.pdf, a dialog prompts you with basic information about sample files contained in the portfolio. Clicking on the "Install Folder-level JavaScript Samples" button and follow the steps to copy JavaScript samples to your user's `JavaScript` folder. When you restart Acrobat you should see a new item Presentation Monitor under the Edit/Acrobat SDK JavaScript menu. But before clicking the menu item, open a PDF file for the presentation (it is generally a PDF file including up to 30 slides). Now click the menu item to get a dialog box to enter the number of minutes you plan to use for the presentation. After that the monitor shown in the top of the slide page will start. When you go through the pages, check the following tools:

- A message showing number of pages untouched.
- A message showing time left.
- A time progress bar.

- A set of page icons:
  - The page icons with the different colors can indicate which page is current, and which pages have been visited.
  - When the mouse enters/exits a page icon, the page image will be shown/hidden in the top left corner.
  - Click a page icon to go to that page.
- A check box (the second one in the top-right corner): check/uncheck to show or hide the time bar and page icons.
- A quit button with "X" (in the top-right corner): click to quit the monitor tool.

Note: While the Thermometer is available in Adobe Reader, this sample will not run there due to use of methods not available.

PresentationNote

Location

JavaScriptSupport/Acrobat SDK JavaScript Samples Portfolio.pdf/Home/JavaScript Samples/Non-Embedded JavaScript/

Description

Creates a temporary note on top of the front PDF file to show the amount of time before a presentation begins.

When you open the Acrobat SDK JavaScript Samples Portfolio.pdf, a dialog prompts you with basic information about sample files contained in the portfolio. Clicking on the "Install Folder-level JavaScript Samples" button and follow the steps to copy JavaScript samples to your user's `JavaScript` folder. When you restart Acrobat you should see a new item Presentation Note under the Edit/Acrobat SDK JavaScript menu. But before clicking the menu item, open the PDF file containing the presentation. Then click the menu item to get a dialog box. Enter the number of minutes before the start of the presentation and click the OK button to close the dialog box. The amount of time until the presentation begins will display and the time will be constantly updated until the specified time period is over. The display will last 10 seconds more after the end time, then go away. You may click the menu item again at any time to stop and remove the display.

RunMediaPlayers

Location

JavaScriptSupport/Acrobat SDK JavaScript Samples Portfolio.pdf/Home/JavaScript Samples/Multimedia/

Description

This Acrobat multimedia sample demonstrates the use of JavaScript to cue up two media players and then start them playing simultaneously.

Click the Play button to see the two movies playing. The movies have no sound. When you click the Play button, JavaScript code will open each player and install an afterReady event listener in each one. When all players have reported afterReady, the code calls the play() method on each player. This way, the players start within a fraction of a second of each other.

System Requirements. Acrobat 6.0 or later. The machine should be ready for multimedia play - sound card, speaker, proper system settings, and a multimedia player, such as Windows Media Player, Apple QuickTime, or Adobe Flash.

ScriptEvents

Location

JavaScriptSupport/Acrobat SDK JavaScript Samples Portfolio.pdf/Home/JavaScript Samples/Multimedia/

Description

This Acrobat multimedia sample demonstrates JavaScript commands sent from a Flash movie. This is a sample showing how to write JavaScript event listener functions to send out movie commands from a movie object, and interpret movie commands as you wish.

Click on the movie in the upper-right box, and the movie will begin to play and write scripts to the bottom script window.

Main functions are set as document-level JavaScript. An AfterScript listener function was created to send out the movie command with a parameter which is the movie script.

SilentPrint

Location

JavaScriptSupport/Acrobat SDK JavaScript Samples Portfolio.pdf/Home/JavaScript Samples/Non-Embedded JavaScript/

Description

Folder level JavaScript code to demonstrate the print APIs. It can be used in Adobe Reader as well as Acrobat. There are many print options that users may set up in the JavaScript print parameters. See the *JavaScript for Acrobat API Reference* and *Developing Acrobat Applications Using JavaScript* for further information about silent print functionality.

Usage

When you open the Acrobat SDK JavaScript Samples Portfolio.pdf, a dialog prompts you with basic information about sample files contained in the portfolio. Clicking on the "Install Folder-level JavaScript Samples" button and follow the steps to copy JavaScript samples to your user's `JavaScript` folder. When you restart Acrobat you should see a new item Silent Print under the Edit/Acrobat SDK JavaScript menu. Click on it to print the current document to the default printer without displaying the Print dialog box.

StoreFormData

Location

JavaScriptSupport/Acrobat SDK JavaScript Samples Portfolio.pdf/Home/JavaScript Samples/Embedded JavaScript/

Description

Demonstrates that a searchable database can be embedded inside a PDF form document, using the Doc's data object methods and other objects and methods. The code in this sample is located inside a PDF document, but it can be modified to be folder-level code to work with other PDF form files.

Usage

Some form data are already embedded in the PDF document, and you can click a form data entry under the Form Data List bookmark to retrieve the data. You can delete a bookmark to remove the data entry. There are buttons for you to reset form fields, to add or modify a set of data, and to search for certain form data. Click the Help button to get to the second page for detailed instructions.

Note: Will not run in Adobe Reader.

TextExtract

Location

JavaScriptSupport/Acrobat SDK JavaScript Samples Portfolio.pdf/Home/JavaScript Samples/Non-Embedded JavaScript/

Description

A folder level JavaScript sample that demonstrates how to extract the text in PDF page contents and save it to a local file. The JavaScript Doc.getPageNthWord method is used to get the words one by one from the current page, and then a data object is created and exported.

Usage

When you open the Acrobat SDK JavaScript Samples Portfolio.pdf, a dialog prompts you with basic information about sample files contained in the portfolio. Clicking on the "Install Folder-level JavaScript Samples" button and follow the steps to copy JavaScript samples to your user's `JavaScript` folder. When you restart Acrobat you should see a new item Extract Text under the Edit/Acrobat SDK JavaScript menu. When the new menu item is clicked, you can select a file to where you can save the text extracted from the current page. You can use Microsoft Word to view the text file.

Note: Requires Reader-enabled PDF document to run in Adobe Reader.

TwoPartInvention

Location

JavaScriptSupport/Acrobat SDK JavaScript Samples Portfolio.pdf/Home/JavaScript Samples/Multimedia/

Description

Demonstrates how to use script events to synchronize UI and multimedia playback. This is a PDF document with sheet music you can click to play back. The music is a QuickTime file which contains a MIDI track. A C++ program using the QuickTime API was used to add a marker at each measure in the QuickTime file, and a script event at each measure and beat within a measure. When the music is playing, the script events trigger JavaScript code in the PDF document that outlines the current measure and beat and turns the page when needed. You can also click a measure for the music to jump to.

You can click About to learn the implementation details.

System Requirements. Acrobat 6.0 or later. The machine should be ready for multimedia play - sound card, speaker, proper system settings, and a multimedia player, such as Windows Media Player, QuickTime, or Flash.

Note: Will not run in Adobe Reader.

3 Mac OS - Interapplication Communications

DistillerControl

Location

IAC/mac/AppleScripts

Description

Demonstrates two methods for converting PostScript files into PDF using the Distiller AppleScript interface. The first uses the default settings and the second sets a specific JobOptions setting.

ObjectProperties

Location

IAC/mac/AppleScripts

Description

Demonstrates how to work with various objects exposed through the Acrobat AppleScript interface including the application, document windows, documents, and pages.

PrintPage

Location

IAC/mac/AppleScripts

Description

Prompts the user to browse for a PDF file then prints the first page to the default printer.

RotatePages

Location

IAC/mac/AppleScripts

Description

Demonstrates low-level page manipulation. It also demonstrates processing all PDF files within a chosen folder.

The user is prompted to select a degree of rotation before browsing for a folder. For each PDF document in that folder, every page in the document is rotated by the specified amount.

SelectText

Location

IAC/mac/AppleScripts

Description

Demonstrates how to create text selections within a document open in the viewer.

WatermarkJsoAS

Location

IAC/mac/AppleScripts

Description

Demonstrates how to access the JavaScript object to execute addWatermarkFromText and addWatermarkFromFile methods. This sample mimics the Windows IAC sample WatermarkJsoVB.

Input files present in the package needs to be placed at desktop. After executing the script, users are provided option to choose an output folder in which to save the stamped file.

4 Windows - Interapplication Communications

AcrobatActiveXVB

Location

IAC/win/VBSamples/AcrobatActiveXVB

Description

Demonstrates how to use the Acrobat ActiveX control to embed a PDF document window with toolbars in a VB.NET application. It also shows how to use the ActiveX APIs to create user graphical interfaces to externally control the PDF window.

Acrobat provides a COM-based automation interface (AcroPDF.dll). You can add it to your project's references to add the control to your toolbox. Once you drag the ActiveX control to a window in your application, you can open a PDF document window with tool bars. You can also use the APIs provided by the ActiveX control to programmatically manage the window for opening a PDF file from a URL or a local disk, page navigation, setting display mode or style, hiding tool bars, and so on.

Usage

You must have Acrobat or Adobe Reader installed to run this VB.NET application. You can access a PDF document in a URL or open a PDF document on a local disk, and then use the tool bar in the ActiveX control or the buttons and dialog boxes implemented with APIs to control the PDF document window.

AcroPDFInHTML

Location

IAC/win/HTMLSamples/AcroPDFInHTML

Description

A simple example of using the PDF control embedded in an HTML page. The <OBJECT> tag with a unique classid for AcroPDF ActiveX control is used to embed it. That is, classid="clsidCA8A9780-280D-11CF-A24D-444553540000".

This sample also demonstrates how to use the automation API on the control from client-side JavaScript.

Usage

Users must have Internet Explorer installed on their machine. To make the sample work properly, set the option to allow blocked content. For example, the user can click the security bar.

The sample includes four HTML files. Click the frameset1.htm file to start. A web page with frames will display in the browser. Input a URL to a PDF file and click the Go button to pop up the PDF ActiveX control. There are two buttons in the left frame, the user can click to move to the previous or next page. The sample code only works with Internet Explorer on Windows, but the approach can be adapted for other browsers.

ActiveViewVB

Location

IAC/win/VBSamples/ActiveViewVB

Description

(Adobe Acrobat only. Does not work on Adobe Reader.) Demonstrates how to use the Acrobat automation interface to display an interactive view of a PDF document within a VB.NET application window.

Usage

This sample is an MDI application that opens PDF documents as if the application were rendering them itself. The user can perform a number of operations to edit the PDF document and manipulate the view.

ActiveViewVC

Location

IAC/win/CSamples/ActiveViewVC

Description

(Adobe Acrobat only. Does not work on Adobe Reader.) Demonstrates how to use the Acrobat automation interface to display an interactive view of a PDF document within a VC.NET application's window.

Usage

ActiveView is an MDI application that opens PDF documents as if the application were rendering them itself. The user can perform a number of operations to edit the PDF document and manipulate the view.

AdobePDFSilentVB

Location

IAC/win/VBSamples/AdobePDFSilentVB

Description

This Visual Basic sample demonstrates how to silently print to the Adobe PDF printer. It processes files from a specified input directory (the default is the application's current directory) and places output PDF files in a specified output directory (the default is the current directory). If a valid directory is not specified, output is directed to My Documents. Output file names are created from the original file name with a .pdf extension.

The application uses Windows printing and can process file types for which there is a registered application, that is, Print appears on the file's context menu. Note that the application will try to process a file that has an associated application for opening the file even if it does not have a print process associated with it. That is, the file's context menu has an Open item but not a Print item. This may generate an unhandled exception. To manage such occurrences, there is an array of file types (file extensions) not to process. The array is initially set to ignore three file extensions: .dll, .exe and .pdf. Others may be added. Files for which an associated application cannot be found are skipped (not converted) without notice.

Because this application is dependent on Windows printing and specific registry entries, successful conversion of specific files, types and locations will vary from machine to machine and user to user. Careful testing should be done to ensure desired results can be achieved.

Usage

The application (AdobePDFSilent.exe) can be invoked from the command line. One or two arguments are required for automated use. If one argument is specified, it is used for both the Input and Output directory paths. If two arguments are included, the first is used to set the Input directory path and the second is used for the Output directory.

To prevent the created PDF documents from opening an Acrobat viewer, unselect the View Adobe PDF Results option in the Adobe PDF print driver printing preferences.

BasicIacCS

Location

IAC/win/C#Samples/BasicIacCS

Description

(Adobe Acrobat only. Does not work on Adobe Reader.) A simple sample that provides the minimum code to use the interapplication communication in a C# application. It includes code to launch Acrobat, open a PDF file (`TwoColumnTaggedDoc.pdf` in the `IAC\TestFiles` directory), and get simple information (the number of pages).

BasicIacJsoVB

Location

IAC/win/VBSamples/BasicIacJsoVB

Description

This simple Visual Basic sample demonstrates how to use the interapplication communication (IAC) JavaScript object in Visual Basic applications. It includes the minimum code to create IAC objects and use their properties and methods. The program opens a PDF file (`IAC\TestFiles\TestForm.pdf`) and gets some information about the document (number of pages, number of words, and number of form fields).

BasicIacOCXCS

Location

IAC/win/C#Samples/BasicIacOCXCS

Description

Demonstrates how to use the PDF ActiveX control in a C# application.

Usage

This sample displays two PDF windows within the sample form. Type in a URL or browse a local PDF file by clicking a Browse button. The file specified in the Address field will be displayed in the corresponding PDF window after you click Go. The two windows can display the same or different PDF files.

BasicIACVB

Location

IAC/win/VBSamples/BasicIacVB

Description

(Adobe Acrobat only. Does not work on Adobe Reader.) A simple Visual Basic sample for interapplication communication. It includes the code to launch the Acrobat viewer, open a PDF file (`IAC\TestFiles\sample.pdf`), and get simple information (number of pages).

BasicIacVC

Location

IAC/win/CSamples/BasicIacVC

Description

Provides the minimum code to use interapplication communication (IAC) in a Visual C++ application. The sample first creates an Acrobat IAC PDDoc object, then tries to open a sample file `C:\simple.pdf`. If the file is opened successfully, it gets the number of pages and displays this number in the dialog box. Otherwise, it shows an error message.

DdeOpenVC

Location

IAC/win/CSamples/DdeopenVC

Description

DdeOpenVC demonstrates how to communicate with Acrobat through DDE interfaces.

Usage

The sample attempts to open a PDF document in Acrobat. The path `"c:\simple.pdf"` is sent to Acrobat as the parameter of the FileOpenEx command. That file needs to exist.

Implementation details

DDE server executables must be running before external applications can initiate a conversation with them. The sample checks the following key in the registry to determine if Acrobat has been installed:

```
HCLM\Microsoft\Windows\CurrentVersion\App Paths\Acrobat.exe
```

If the key is found, the path stored under the key is launched.

DistillerCtrlVB

Location

IAC/win/VBSamples/DistillerCtrlVB

Description

Demonstrates how to use the Acrobat Distiller automation interface to convert PostScript files to PDF in a VB.NET application.

Usage

Click Select Input File to select the PostScript file that will be processed. The Visual Basic implementation also allows users to select a job options file to use when processing the file. The progress of the conversion operation is shown in the application window.

Implementation details

Both implementations receive progress updates from Distiller. This is achieved through the use of the _PdfEvents IConnectionPoint interface that the automation interface supports.

Visual Basic makes use of this interface rather easy - simply declare the Distiller application object with the WithEvents keyword, for example, `Private WithEvents pdfDist As PdfDistiller`

When the WithEvents keyword is used, Visual Basic provides the framework for the application to provide implementations for each event in the interface. The C++ implementation is more involved. The application must provide an implementation of the _PdfEvents interface and register it with Distiller using the Advise method on the _PdfEvents IConnectionPoint interface.

DistillerCtrlVC

Location

IAC/win/CSamples/DistillerCtrlVC

Description

(Adobe Acrobat only. Does not work on Adobe Reader.) Demonstrates how to use the Acrobat Distiller automation interface to convert PostScript files to PDF in a C++ application.

Usage

Click Select Input File to select the PostScript file that will be processed. There is one in IAC\TestFiles that you can use. The Visual Basic implementation also allows users to select a job options file to use when processing the file. The progress of the conversion operation is shown in the application window.

Implementation details

Both implementations receive progress updates from Distiller. This is achieved through the use of the _PdfEvents IConnectionPoint interface that the automation interface supports.

Visual Basic makes use of this interface rather easy - simply declare the Distiller application object with the WithEvents keyword, for example, `Private WithEvents pdfDist As PdfDistiller`

When the WithEvents keyword is used, Visual Basic provides the framework for the application to provide implementations for each event in the interface. The C++ implementation is more involved. The application must provide an implementation of the _PdfEvents interface and register it with Distiller using the Advise method on the _PdfEvents IConnectionPoint interface.

DistillerCtrlWMVC

Location

IAC/win/CSamples/DistillerCtrlWMVC

Description

(Adobe Acrobat only. Does not work on Adobe Reader.) Demonstrates how to use the Acrobat Distiller Windows messaging interface to convert PostScript files to PDF in a C++ application.

Usage

The sample allow users to select the PostScript file that will be processed. The user can also specify the level of interactivity that Distiller should use to determine the name of the output file.

Implementation details

An instance of the Distiller application must be running to receive the messages from the external application. The sample checks the following key in the registry to determine if Acrobat has been installed:

```
HCLM\Microsoft\Windows\CurrentVersion\App Paths\AcroDist.exe
```

If the key is found, the path stored under the key is launched.

External applications can initiate conversion operations within Distiller by sending it the WM_COPYDATA message. As the LPARAM parameter, you must pass a pointer to a COPYDATASTRUCT structure which contains (amongst other things) a DISTILLRECORD structure. The DISTILLRECORD structure contains the conversion parameters for the operation being performed.

For the WPARAM parameter, you can pass an HWND to which Distiller should report the status of each conversion operation. This response is also passed using a WM_COPYDATA message. When the specified HWND receives the reply message from Distiller, the LPARAM parameter contains a pointer COPYDATASTRUCT structure which contains a DISTILLRECORD structure. The param member of this structure will contain zero if the operation was completed successfully, or -1 if some error occurred.

ExecuteScriptIacVB

Location

IAC/win/VBSamples/ExecuteScriptIacVB

Description

Demonstrates how to execute JavaScript code by calling the AcroForm OLE ExecuteThisScript method in Visual Basic applications. The default code writes information about the active PDF document to a new PDF report file, then opens the report. The program provides a dialog box for users to modify, rewrite, and execute their own JavaScript code.

Usage

- Click PDF Document to open or change the active PDF file open in Acrobat.
- Click Execute to execute the JavaScript code. First try the default JavaScript code mentioned above, and check the PDF report in Acrobat. Close the report.
- Click Clear to clear the text window, then input your JavaScript to execute.
- Click Reset to restore the default JavaScript code.
- Click Help to check the help message.
- Click Close to quit the program. Acrobat viewer also quits if it is opened by this program or it has no PDF files opened.

The BasicIacVBJso sample also shows how to execute JavaScript code from an IAC application.

FillFormCS

Location

IAC/win/C#Samples/FillFormCS

Description

Demonstrates how to fill a form from a C# application using interapplication communication.

Usage

Run the application.It uses the PDF file in IAC\TestFiles\SampleForm.pdf. The form field Name will be filled with value `John Doe`.

FormsAutomationVB

Location

IAC/win/VBSamples/FormsAutomationVB

Description

Demonstrates how to create and manipulate form fields using the automation interface exposed by the Acrobat Forms plugin in a VB.NET application.

Usage

Before running the sample you must copy the `FormsAutomation.pdf` file to the `C:\` directory. The sample creates a variety of form fields in the file. It is best to have Acrobat open and visible to see the form creation.

To run as a batch process, save the form and close the file in Acrobat. There is commented-out code in the sample that shows how to do this.

JSObjectAccessVB

Location

IAC/win/VBSamples/JSObjectAccessVB

Description

(Adobe Acrobat only. Does not work on Adobe Reader.) Demonstrates how to access the JavaScript object model through the Acrobat automation interface.

The sample opens a PDF (IAC\TestFiles\FormSample.pdf) with a template that contains form fields, loads data from a data file (IAC\TestFiles\data.txt), and then saves the resulting PDF in a outputfiles directory at the root level of the sample.

JSObjectControlCS

Location

IAC/win/C#Samples/JSObjectControlCS

Description

(Adobe Acrobat only. Does not work on Adobe Reader.) The sample includes functionality to use the JSObject interface. After opening a document, you can affix a watermark, add form data to copies of the document, and/or search the document for a word.

A sample watermark stamp and files for testing forms are included in the IAC\TestFiles directory. These are RecievedStamp.pdf which can be used as a watermark, and FormSample.pdf and data.txt for the form testing.

JSOFindWordVB

Location

IAC/win/VBSamples/JSOFindWordVB

Description

(Adobe Acrobat only. Does not work on Adobe Reader.) A JSObject sample that shows how to access JavaScript using Visual Basic. The approach is to get the JavaScript object from the PDDoc of a PDF file, then call JavaScript methods.

The application displays a dialog box in which the user can select a PDF file and input a word (in Roman characters only) to find. Clicking the Find Word button displays the page on which the word is first found and highlights the word. After each occurrence is found, the user is asked whether to continue the search. The final count of words found is displayed in the dialog box.

Implementation details

Since the JavaScript code runs slowly, it is not suitable for searching through a large PDF file.

RemoteControlAcrobatVC

Location

IAC/win/CSamples/RemoteControlAcrobatVC

Description

An MFC Windows program that controls Acrobat remotely through IAC (interapplication communication). It shows how to start IAC, create IAC objects, and call their methods. The program has an Acrobat menu that has the following items:

Launch

- App submenu: Show Acrobat, Hide Acrobat
- AVDoc submenu: Open, Close, Print PDF, Find Text in PDF
- AVPageView submenu: Display Page Number, Go to Next View, Go to Previous View, Go to Page
- Exit

The menu item Find Text in PDF simply finds and highlights the first occurrence of a word in Roman characters. Since a workspace with sample code has been set up, the user can easily add any other Acrobat IAC objects and methods to the program. The source file `RemoteControlAcrobat.cpp` contains information about the program, its limitations, and how to extend it in real programs.

Note: The other menus do not have handlers created for them, and so don't do anything.

SearchPdfVB

Location

IAC/win/VBSamples/SearchPdfVB

Description

Demonstrates how to communicate with the Search plugin through its DDE interface. The DDE interface allows external applications to manage indices and initiate queries.

Usage

The SearchPDF application divides its functionality into two distinct operations; index manipulation and search queries. Users can add, remove, enable, or disable index files.

Users can also formulate a search query and specify a number of query options, including the query parser to be used and the maximum number of documents returned.

Implementation details

The application uses a C-based DLL to manage the DDE conversation with the Search plugin. The interface provided by the DLL mimics the functionality exposed by the plugin's DDE interface. The DLL (`ddeproxy.dll`) should be in the system directory or the directory where the Visual Basic program is located.

StaticViewVB

Location

IAC/win/VBSamples/StaticViewVB

Description

(Adobe Acrobat only. Does not work on Adobe Reader.) Demonstrates how to use Acrobat's automation interface to render the contents of a PDF page into a VB.NET application's window.

Usage

StaticView is an MDI application that opens PDF documents as if the application were rendering them itself. The user can manipulate the view of the PDF document through the toolbar and View menu.

StaticViewVC

Location

IAC/win/CSamples/StaticViewVC

Description

(Adobe Acrobat only. Does not work on Adobe Reader.) Demonstrates how to use the Acrobat automation interface to render the contents of a PDF page into a C++ application's window.

Usage

StaticView is an MDI application that opens PDF documents as if the application were rendering them itself. The user can manipulate the view of the PDF document through the toolbar and View menu.

Note: When protected mode is enabled, the PDF opens in a temporary window.

WatermarkJsoVB

Location

IAC/win/VBSamples/WatermarkJsoVB

Description

A VB.NET sample that shows how to add watermarks to a PDF document programmatically. Two JavaScript methods, addWatermarkFromFile and addWatermarkFromText, are used through the JavaScript object in the IAC application.

Usage

The program will open IAC\TestFiles\SamplePDF01.pdf and add IAC\TestFiles\Stamp.pdf as a water mark. It will save the resulting file in the root directory of the sample. A stamp with a date and time will be added to the top left corner of the first page of the PDF document.

Due to security restriction enforced by Vista, exercise this sample in one of the two ways on Vista: (1) Run Acrobat with "administrator" privilege, then run the sample with "administrator" privilege. (2) Make sure Acrobat is not running. Quit Acrobat if it is. Run the sample with "administrator" privilege. To run an application with administrator privilege, right-click on the application icon to bring up the context menu then select "run as administrator."

Note that logging in with administrator capacity does not override the security restriction.

5 Navigator Samples

Navigators

Location

Navigators

Description

These samples get you started with Navigator development. Other than the TileView sample, the samples serve as snippets that address only specific features they demonstrate.

- Starter contains only the required property INavigator.set host() and some functions to take care of the race condition when a Navigator is initialized. It can serve as a framework for your Navigator development.
- ReadWriteResources is a sample that demonstrates how to add resources to a portfolio and how to read and write the contents of the navigator custom resource that exists for the given path.
- ApplyColorPalette demonstrates how to apply a suggested palette of colors for the navigator.
- GetLocalizedString shows how to use localized string in navigators.
- TileView is a navigator that has folder navigation and uses the Acrobat API to feed thumbnails to an item renderer. You can use the context menu to view or manipulate the portfolio. Drag and drop only support drag into the current directory.
- TrivialDragDrop demonstrates how to drag and drop to the current directory of the navigator.
- DragOut extends TrivialDragDrop sample to support drag from portfolio to the OS.
- DragWithin extends DragOut sample to support drag into sub-folders.
- CheckPermission demonstrates how to set certain functionalities of a navigator based on permissions granted.
- SimpleSort demonstrates how to set and get sorting in a navigator.

Usage

Switch workspace and point to SDK sample NavigatorSupport directory.

Import the Samples to your workspace.

The Navs folder contains the .nav files for the samples and their corresponding folders containing resouces needed to build the .nav files. You can use your favorite zip tool to modify the content of the .nav file.

When it is time for you to choose a layout, select SDK Sample Layouts. All the SDK sample Navigators are under that category.

6 Tools

Plugin Wizard

Location

Acrobat SDK/Visual Studio App Wizard

Description

Wizard to create plugins for Acrobat with Visual Studio.

The Plugin Wizard extends Microsoft Visual Studio to simplify the creation of Visual Studio C++ projects that create plugins for Acrobat.

Installation

If Visual Studio 2019 is installed when the Acrobat Plugin wizard installer is run, the installer will install the necessary files in the Visual Studio installation.

Usage

In Visual Studio, create a new project by clicking on **New Project** or choosing the File > New > Project command. Select "Visual C++ Projects" in the Project Types pane. Choose Acro9PIWiz in the Templates pane. Enter a name and location for the project and click OK.

The first time you use the Plugin wizard, you will be prompted to locate Acrobat SDK header path. You can check "Do not show at startup" if you don't want to see the dialog next time you start Visual Studio. However, if you want to check whether you have specify the correct location for the headers at a later time, you can always click on the "Headers" button to bring up the dialog showing the header path.

The Plugin Wizard will now start, offering you many high-level choices about the plugin that will be created. Select those that you desire and the wizard will create the project and populate it with the necessary files.

Replaceable functions can have many different return types, from void to built-in to structured types. Because of this, projects that use replaceable functions will draw compiler warnings as the variable used to hold the return value is not initialized. You should, of course, complete the replaceable function to do what you would like, and in the process initialize the return value to eliminate the compiler warning.

Limitations

- Requires Visual Studio 2019
- Windows only
- If you intend to use Visual Studio App Wizard to create a plugin that uses privileged operations and you are running on Windows Vista, you must establish yourself as an administrator. This is required even if your user ID has Admin privileges.

- Select the Windows start icon.From All Programs, navigate to Visual Studio C++. Right click Visual Studio C++.Select "Run as ...."Select "Run as administrator", which is the only selection that elevates your privilege level to administrator.

ShowPermissions

Location

Acrobat SDK/ShowPermissions

Description

ShowPermissions uses the PDDocPermRequest function to check operation permissions for the front PDF, and saves the results to text files.

Since the Acrobat products have different versions and variations and the PDF may have certain level of Reader-Enabled usage-rights, this plugin can provide useful information about the Acrobat functionality / API availabilities and the PDF permitted operations.

ShowPermissions.api will work with Acrobat 7 or later, and it has been Reader-enabled for use with Adobe Reader. Note that if you directly build the ShowPermissions plugin sample in the SDK, then that plugin won't be able to run in Adobe Reader.

Usage

Copy this file to the Acrobat or Adobe Reader plug_ins directory. The plugin will add a menu item titled "Show Permissions" under the Tools menu. When chosen, you can select a folder for the output, and then save the operation permission list to a plain text file and a tab delimited text file. The latter can be opened by Excel. The output's filenames are the PDF filename plus -permissions.txt or .xls.

Note: Windows only.