

ADOBE® INDESIGN® CS5.5



GETTING STARTED WITH THE ADOBE INDESIGN CS5.5 SERVER SDK



© 2011 Adobe Systems Incorporated. All rights reserved.

Getting Started with the Adobe® InDesign® CS5.5 Server SDK

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

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

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

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

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

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

Document Update Status

CS5.5 Unchanged Content not guaranteed to be current.

Contents

1	Getting Started	5
	First Step	5
	Next Steps	5
	Samples	5
	InDesign Server SDK Footprint	6
	/docs/guides/	6
	/docs/references/	6
	/external/	7
	/legalnotices/	7
	/lib/	7
	/samples/	7
	/scripting/	7
	/tools/	7
2	SOAP Samples	8
	Introduction	8
	DropBox-flex-soap	8
	Usage	8
	Configuration	8
	Running Flex Builder 3	9
	Using amxmlc	9
	sampleclient-java-soap	9
	Usage	9
	Running from the command line	10
	Building with ant	10
	Using Eclipse	11
	Build warnings	12
	sampleclient-aspnet-soap	12
	Getting the sample to work under IIS	12
	Building the client using Visual Studio 2008	14
	ASP.NET implementation	14
	Troubleshooting	15
	sampleclient-csharp-soap	16
	Usage	16
	Building with Microsoft Visual Studio 2008	17
	Debugging	17
	RunScript results	18
	sampleclient-flex-soap	18
	How to build the sample	18
	Usage	18
	Troubleshooting	19
	sampleclient-php-soap	20

	Requirements	20
	Install the sample under Apache or IIS	21
	Using the sampleclient	21
	Troubleshooting	22
3	Java/CORBA Samples	23
	Introduction	23
	idsp-wsdl-java	23
	Building with Ant	23
	Using Eclipse	23
	Build warnings	24
	sampleclient-java-corba	24
	Usage	25
	Running from the command line	25
	Building with Ant	25
	Using Eclipse	26
	Running from within Eclipse	26
	Java language warnings	26
	snippets	27
	Usage	27
	Running from the command line	27
	Building with Ant	28
	Using Eclipse	28
	Running from within Eclipse	28
	Java language warnings	29
	snippets-runner	29
	Quick start	29
	Usage	30
	Running from the command line	30
	Building with Ant	30
	Using Eclipse	31
	Running from within Eclipse	31
4	COM Samples	32
	Introduction	32
	helloworld-vb-com	32
	About InDesign Server COM and Visual Basic	32
	Usage	33
	Building with Microsoft Visual Studio 2008	33
	Debugging	34
	Output	34
	helloworld-csharp-com	34
	Usage	34
	Building with Microsoft Visual Studio 2008	34
	sampleclient-csharp-com	35
	Usage	35
	Building with Microsoft Visual Studio 2008	36
	Debugging	36
	Script results	36

1 Getting Started

The Adobe® InDesign® Server Software Development Kit (SDK) contains information about using CORBA and SOAP to interact with InDesign Server. A primary goal of the SDK is to document and demonstrate the InDesign Server Java/CORBA API. The SDK also contains samples demonstrating SOAP interaction through several technologies.

After you have installed InDesign Server, and after learning to run InDesign Server and using SOAP or Java to run a sample provided in the SDK, as described in this document, you may want to extend the capabilities of InDesign Server. You can do this by writing scripts, plug-ins, or even applications that integrate InDesign with a publication workflow.

First Step

Your first step in using the SDK is to read *Introduction to Adobe InDesign CS5 Server Development* (intro-to-indesign-server.pdf). It describes how to install and run InDesign Server in a simple environment, and briefly describes how to communicate with InDesign Server through both SOAP and the Java/CORBA API.

Next Steps

- ▶ If you want to write a Java component to interact with InDesign Server, begin by reading “Working with InDesign Server Java” in *Adobe InDesign Server Solutions*. It gives an overview of the Java API, and contains many example code listings. You also can use the InDesign Server Java API reference and the sample code in the <SDK>/samples/ folder.
- ▶ If you want to send a script to InDesign Server using SOAP, begin by reading “Working With InDesign Server SOAP” in *Adobe InDesign Server Solutions*. The samples folder contains SOAP projects using several technologies, including C#, ASP.NET (VB and C#), PHP, Flex, and COM (VB and C#).
- ▶ If you are interested in analyzing the scalability and performance of InDesign Server for your workflow, begin by reading “InDesign Server Scalability and Performance” in *Adobe InDesign Server Solutions*. The tools folder contains analysis tools discussed in the documents.

Introduction to Adobe InDesign Server SDK Development lists additional resources.

Samples

The last chapters of this document provide descriptions of the SOAP, Java/CORBA, and COM samples that are included with the InDesign Server SDK.

InDesign Server SDK Footprint

/docs/guides/

This folder contains documentation that describes how to use InDesign Server via CORBA and SOAP, and the InDesign Server Java API. It contains the following files:

- ▶ *getting-started.pdf* — This document, *Getting Started With the Adobe InDesign Server SDK*:
 - ▷ Lists and describes the documentation and folders that are provided with the InDesign Server SDK.
 - ▷ Discusses the samples included with the SDK. It covers the design of each sample and how to install and run the sample.
- ▶ *intro-to-indesign-server.pdf* — This document, *Introduction to Adobe InDesign CS5 Server Development*, gives an overview of InDesign Server and how to launch it for use with SOAP and/or CORBA. It also discusses how to use the sample SOAP client, *SampleClient*, and how to run an InDesign Server Java component.
- ▶ *server-solutions.pdf* — This document, *Adobe InDesign Server Solutions*, includes:
 - ▷ “Working with InDesign Server Java,” which gives an overview of the InDesign Server Java API. It covers special elements of the API and how to write, compile, and run Java code written to the API. The document also contains many sample code listings.
 - ▷ “Regenerating the InDesign Server Java API,” which gives instructions for regenerating the *InDesignServerAPI.jar* file. If you create an InDesign Server plug-in using C++, you need to do this to make your plug-in’s functionality accessible from Java.
 - ▷ “Working with InDesign SOAP,” which gives an overview of the InDesign Server SOAP implementation. It describes the InDesign Server WSDL, InDesign Server SOAP request and response XML format, and the sample client implementations provided in the SDK.
 - ▷ “Working With Load Balancing and Queueing,” which contains information about Load Balancing and Queueing (LBQ), an add-on component for Adobe InDesign Server. LBQ provides load balancing across multiple instances of InDesign Server on one or more servers.
 - ▷ “Scalability and Performance,” which discusses tools and methods you can use to analyze your system’s performance and scalability with regard to InDesign Server. It also publishes benchmarking results and gives instructions on how to benchmark your own InDesign Server system.

/docs/references/

This folder contains a reference to the InDesign Server Java API and other, supporting files:

- ▶ *errorcodes.htm* — This contains InDesign C++ API error codes.
- ▶ *IDSP.wsdl* — This is the WSDL file used to describe the InDesign Server SOAP interface.
- ▶ *InDesignServerAPIRef.chm* — For Windows®. This is a compiled HTML Help file. It is a text-searchable help file that contains reference documentation for all public APIs included in this SDK. To view the contents, simply double-click the file’s icon.

- ▶ *InDesignServerAPIRef.tar.gz* — For Mac OS®. This archive file contains HTML reference documentation for all public APIs included in this SDK. To decompress the file, double-click the file's icon. It decompresses to an "InDesignServerAPIRef" subfolder. To navigate the HTML reference documentation set, open the index.html file in the InDesignServerAPIRef sub-folder.

/external/

This folder contains non-Adobe libraries required by sample projects.

/legalnotices/

This folder contains the End User License Agreement and Legal Notices.

/lib/

This folder contains the InDesign Server Java API jar file, *InDesignServerAPI.jar*. This Java JAR file contains the InDesign Server Java API code. When writing your InDesign Server Java code, you import classes from this JAR file.

/samples/

This folder contains sample InDesign Server projects demonstrating the use of the Java/CORBA API, the SOAP API, and other useful technologies. Each sample is contained in its own folder. For information on the samples, see *Introduction to Adobe InDesign CS5 Server SDK Development* in the docs/guides folder. Each of the Java/CORBA samples has its own Eclipse project file and Ant build file.

/scripting/

This folder contains documentation on how to write scripts for InDesign Server. It also contains many sample InDesign Server scripts written in JavaScript, AppleScript (Mac OS only), and VBScript (Windows only).

/tools/

This folder contains scripts and other tools used by Adobe to benchmark InDesign Server to investigate the scalability and performance of InDesign Server. The tools contained in this folder were used to establish the benchmark results published in the "Scalability and Performance" chapter of *Adobe InDesign Server Solutions*. You can use these tools as is or add custom tests, allowing you to analyze the performance of your system.

2 SOAP Samples

This chapter contains information about the SOAP samples contained in the Adobe® InDesign® Server SDK. These samples demonstrate how to interact with InDesign Server using SOAP.

Introduction

All the SOAP samples are clients that send a script to InDesign Server through the `RunScript` method defined in the InDesign Server WSDL. These sample clients are written in a variety of languages:

- ▶ Java — [sampleclient-java-soap](#)
- ▶ ASP.NET (C# and VB) — [sampleclient-aspnet-soap](#)
- ▶ C# — [sampleclient-csharp-soap](#)
- ▶ Flex — [DropBox-flex-soap](#), [sampleclient-flex-soap](#)
- ▶ PHP — [sampleclient-php-soap](#)

For more information on the InDesign Server SOAP implementation, see “Working with InDesign Server SOAP” in *Adobe InDesign Server Solutions*.

DropBox-flex-soap

DropBox is a Flex/AIR application that demonstrates interaction with InDesign Server using Web services. It provides a simple graphical user interface for executing a variety of InDesign Server scripts against files on your computer.

Usage

The DropBox user interface consists of a table of images, or *script widgets*. Each script widget has an InDesign Server script associated with it. Dragging and dropping a file over a script widget causes the InDesign Server script associated with that script widget to be executed against that file. In response, InDesign Server generates a modified file and places it in the same directory as the input file. The filename given to the file is the name of the input file with `out` appended to it.

Configuration

To configure DropBox, click on the wrench icon in the title bar. This opens a window containing four tabs. Each tab has a unique grouping of DropBox configuration settings:

- ▶ **Application** — This tab allows you to configure the look of the application. Specifically, it allows users to specify the size of the images and the number of images in each row and column, in one view. If more script widgets are added than can fit in one view, multiple views are generated. When this happens, arrow buttons appear, allowing you to scroll through all views.
- ▶ **InDesign Server** — This tab allows you to set the URL of the WSDL file for the instance of InDesign Server targeted. Currently, only one instance of InDesign Server can be targeted by this application.

- ▶ **Scripts** — This tab allows you to add, edit, or remove script widgets from the DropBox interface. The images and scripts and associated with each widget can be modified here.
- ▶ **Network** — This tab is used when InDesign Server and the DropBox application are on separate machines. It allows you to specify a common share that can be used by both the DropBox client and InDesign Server, to place and access files. The permissions assigned to the share need to allow InDesign Server and the DropBox clients to read and write to it.

If the DropBox application and InDesign Server run on different operating systems, or if drive mappings are used, there may be different paths to the common share. The Network tab allows you to specify different paths to the common share for DropBox and InDesign Server.

Running Flex Builder 3

1. In the Flex Navigator window, right-click and choose Import...
2. Choose General > Existing Projects into Workspace.
3. Click Next.
4. Choose “Select root directory,” and browse to the `dropbox-flex-soap` folder.
5. Click Finish.

NOTE: The script files in `src/content/scripts` should be set to read/write; otherwise, there will be errors when used with a network share enabled.

Using amxmlc

The compiler is located in the `bin` folder of the Flex SDK installation folder. Open a command line or Terminal window, set the current directory to the `bin` folder, then use the following command (change the path as necessary):

```
amxmlc SDK/samples/dropbox-flex-soap/src/dropbox.mxml
```

This command creates a `bin-debug` folder in the `SDK/samples/dropbox-flex-soap` folder and copies the compiled SWF file to that location.

sampleclient-java-soap

This client application is written in Java and communicates with InDesign Server through SOAP. It uses the `RunScript()` method provided in the `idsp-wsdl-java.jar` file to execute a JavaScript file within InDesign Server. The source code comprises two files:

- ▶ `CommandLineArguments.java` — Handles the parsing of the command-line arguments.
- ▶ `SampleClient.java` — Contains the `main()` method, and controls the flow of the application.

Usage

Syntax `sampleclient -host hostURL scriptPath [-server] [-repeat number] [scriptArgs]`

Parameters	<code>-host</code>	The host to target with the script. The targeted instance of InDesign Server.
	<code>hostURL</code>	A valid URL (using the prefix <code>http://</code>) that specifies the IP address and port of the server. If the client and server are running on the same machine, this can be <code>http://localhost:portID</code> , where <code>portID</code> is a valid port number.
	<code>scriptPath</code>	The full path of the JavaScript file to send to the host.
	<code>-server</code>	Indicates that <code>scriptPath</code> is the name of a JavaScript file on the server's file system. If <code>-server</code> is not specified, <code>scriptPath</code> is the name of a JavaScript file on the client's file system.
	<code>-repeat number</code>	Indicates how many times to execute the script.
	<code>scriptArgs</code>	One or more arguments to pass in to the JavaScript. Each <code>scriptArg</code> is formatted as <code>name=value</code> . Multiple <code>scriptArgs</code> are separated by a space; for example: <code>arg0=100 arg1=200 arg3="this is some text"</code>

Example `SampleClient -host http://localhost:12345 D:\javaScripts\HelloWorld.jsx -repeat 4 -server arg0=100 arg1="this is text"`

NOTE: When launching the sample client, InDesign Server must first be running and configured for use with SOAP using the designated port. To learn how to launch InDesign Server, see *Introduction to Adobe InDesign CS5 Server Development*.

Running from the command line

For your convenience, scripts are provided to execute the sample client. The scripts are located in the project's scripts folder. To execute a script, open a shell window (Command Prompt on Windows or Terminal on Mac OS) and set the current directory to the project's scripts folder, then execute the script providing the appropriate parameters.

For example (replace path names with valid data):

Windows

```
cd "c:\Program Files\Adobe InDesign CS5 Server SDK\samples\sampleclient-java-soap\scripts"
sampleclient -host http://localhost:12345 "c:\myJavaScripts\myscript.jsx" arg0=1
arg1="hello"
```

Mac OS

```
cd "/Applications/Adobe InDesign CS5 Server SDK/samples/sampleclient-java-soap/scripts"
sh sampleclient.sh -host http://localhost:12345 "/myJavaScripts/myscript.jsx" arg0=1
arg1="hello"
```

Output is written to the shell window and/or the InDesign Server console.

Building with ant

1. Open a shell window and set the current directory to the project folder, where `build.xml` is located:

```
cd <SDK>/samples/sampleclient-java-soap
```

2. Execute the ant build script:

```
ant
```

The `ant` build script compiles the source code and outputs a JAR file to the `lib` folder.

Using Eclipse

An Eclipse project file is provided in the project folder. To open the `sampleclient-java-soap` Eclipse project file, run the Eclipse IDE and perform the following steps:

1. Import the project:
 - ▷ From the main menu, choose File > Import... to open the Import wizard.
 - ▷ Choose General > Existing Projects into Workspace, and click Next.
 - ▷ Choose "Select root directory:" and click the associated Browse... button.
 - ▷ In the File dialog, select the root folder of the SDK and click OK.
 - ▷ A list of available SDK projects appears in the Projects window. The `sampleclient-java-soap` project depends on both the external and `idsp-wsdl-java` projects, so make sure to check those two projects as well as the `sampleclient-java-soap` project.
 - ▷ Click Finish. If the required projects do not appear in the Projects window, they probably exist in the workspace.
2. Use Ant to build the project and JAR file.

Right-click on the project's `build.xml` file in the Package Explorer window, and choose Run As > Ant Build. This builds the source code and exports the JAR file.

NOTE: To build with Ant from the command line, make sure you have a `JAVA_HOME` environment variable set up that points to your JDK folder. To build with Ant from within Eclipse, make sure the `<your-JDK-path>/bin` path is specified in the `PATH` variable.

NOTE: To build `sampleclient-java-soap`, make sure `idsp-wsdl-java` had been built successfully.

Running from within Eclipse

1. In the Project Explorer, select the root folder of `sampleclient-java-soap`.
2. From the main menu, choose Run > Run... to open the Run wizard.
3. Click "New launch configuration."
4. In the Name box, enter a name for the launch configuration.
5. Create a new Run Configuration, with the Main class specified as `com.adobe.ids.SampleClient`.
6. Select the Arguments tab, and enter the arguments necessary to run; for example:

```
-host http://localhost:12345 "c:\myJavaScripts\myscript.jsx" arg0=1 arg1="hello"
```
7. Click Run.

After you create the run configuration, then next time you only need to select the configuration and click Run.

Output is written to the Eclipse console and/or the InDesign Server console.

Build warnings

“Unable to find required classes (javax.activation.DataHandler and javax.mail.internet.MimeMultipart). Attachment support is disabled.”

This warning does not affect the ability to run `sampleclient-java-soap`, as attachment support is not a required element. To remove the warning, you need to install Java Mail Framework and JavaBeans Activation Framework, then add `mail.jar` and `activation.jar` to your classpath:

- ▶ `mail.jar` is supplied in the Java Mail Framework. It can be downloaded from <http://java.sun.com/products/javamail/downloads/index.html>.
- ▶ `activation.jar` is supplied in the JavaBeans Activation Framework. It can be downloaded from <http://java.sun.com/javase/technologies/desktop/javabeans/jaf/>.

Java language warnings

The InDesign CS5 Server SDK supports Java 1.4.2, so it does not use Java elements newer than 1.4.2. If your Java version is newer, you may get build warnings like the following:

- ▶ “References to generic type `Vector<E>` should be parameterized.”
- ▶ “The serializable class `class_name` does not declare a static final `serialVersionUID` field of type `long`.”

These warnings can be ignored safely.

sampleclient-aspnet-soap

ASP.NET works in conjunction with Microsoft’s Internet Information Services (IIS) and Visual Studio 2008. If you are new to ASP.NET, you will find valuable information at <http://www.asp.net/>, <http://www.iis.net/>, and <http://www.microsoft.com/visualstudio/en-us/default.aspx>

Developing an ASP.NET Web Site to work with InDesign Server’s SOAP model is fairly straightforward. There are two sample client projects to get you started. They are named `sampleclient-aspnet-vb-soap` and `sampleclient-aspnet-cs-soap` and are located in the InDesign Server SDK `samples/sampleclient-aspnet-soap` folder.

Getting the sample to work under IIS

Your first step toward using the sample client is to get the sample to work under IIS. We assume that you already installed and configured IIS to run on your system.

This sample was written to work with IIS on any Windows platform. Windows nonserver platforms, like Windows XP Professional, do not allow you to add a new site; they only allow you to add a virtual directory to the existing default site. For this reason, adding a virtual directory was used in this sample.

Install the sample under IIS

1. Copy the sample folder, `sampleclient-aspnet-soap`, from the InDesign Server SDK to the `wwwroot` folder of your IIS installation. Typically this is `c:\inetpub\wwwroot`.

2. Open the Internet Information Services snap-in (Control Panel > Administrative Tools > Internet Information Services).
3. Use the tree view to navigate to the sample folder. Typically this is found at MY-MACHINE (local computer) > Web Sites > Default Web Site >. If the folder does not appear, make sure IIS is started (using the `iisreset/START` command) or try using the Refresh button.
4. Open the `sampleclient-aspnet-soap` folder to display the C# and VB sample folders.
5. Right-click the desired sample folder, and select Properties.
6. In the Directory tab of the Properties dialog, click Create. This configures your folder as an application. Click OK to exit the Properties dialog.
7. Test the Web site using a browser. If `wwwroot` is your default Web site, these are the URLs of the sites:

<http://localhost/sampleclient-aspnet-soap/sampleclient-aspnet-cs-soap/Default.aspx>
<http://localhost/sampleclient-aspnet-soap/sampleclient-aspnet-vb-soap/Default.aspx>

These links will display a simple Web page that allows you to specify an InDesign Server host and execute a few sample scripts against InDesign Server.

NOTE: You may get the following error:

```
A name was started with an invalid character. Error processing resource
'http://localhost/sampleclient-aspnet-soap/samplecl...
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="Default.aspx.cs" Inherits="_Default" %>.
```

If you get this error, try to do the following from the command line:

```
>cd C:\Windows\Microsoft.NET\Framework\v2.0.50727
>aspnet_regiis.exe -i
```

Then restart IIS.

Test the client's interaction with InDesign Server

1. Launch InDesign Server for use with SOAP (for example, `indesignserver -port 12345`).
2. Fill in the information on the client's Web form:
 - ▷ **InDesign Server Host** — Enter the URI of the desired InDesign Server instance (for example, `http://localhost:12345`).
 - ▷ **Choose a script** — Select `MakeDocument.jsx`. The listed scripts are read from the `sampleclient-aspnet-soap/IDS_Scripts` folder. Scripts added to the folder are automatically displayed in the list. AppleScript scripts run only on a Mac OS version of InDesign Server, and VBScripts run only on a Windows version of InDesign Server. Error #30485 occurs if you choose an incompatible script.
 - ▷ Create a scriptArg with the name `saveAs` and the value set to a valid path for an InDesign document (for example, `c:\myNewFile.indd`). The path must be based on the file system of the targeted instance of InDesign Server.
 - ▷ Click Call RunScript.

The script is sent to and run by the targeted instance of InDesign Server. The script creates a new, empty InDesign document and saves it to the path specified by the `saveAs` scriptArg. The Results box on the form displays the resulting file path of the created file.

Building the client using Visual Studio 2008

This sample was developed using Visual Studio 2008. To further develop the client using Visual Studio:

1. Run Visual Studio.
2. Choose File > Open > Web Site, browse to the location of the existing project (like `sampleclient-aspnet-cs-soap` or `sampleclient-aspnet-vb-soap`), and click OK.

You will now have a Visual Studio solution file to use for development and debugging.

ASP.NET implementation

Web form

The user interface for the client was developed using a Web form. This Web form contains text-entry boxes allowing the user to enter parameters necessary to make the call to `RunScript`. The Web form sets the `CodeFile` attribute of the Page to hook up a codebehind file. The codebehind file contains the source code (Visualized or C#) that implements the interaction with the SOAP implementation.

SOAP implementation

The SOAP implementation occurs in the `Service` class. The `Service` class is a proxy for the InDesign Server SOAP service. The Service code is generated automatically by adding a Web Reference for the InDesign Server WSDL to the Visual Studio project. This was done using the following steps:

1. Right-click the project in Visual Studio and select "Add Web Reference..."
2. Choose the InDesign Server WSDL by one of these methods:
 - ▷ Choose the "Web Services on the local machine" link, and select `IDSP.wsdl` (`IDSP.wsdl` is located in `<SDK>/docs/references`).
 - ▷ Enter the WSDL URL in the URL text box (for example, `http://localhost:12345/service?wsdl`) and click Go.
3. After Visual Studio loads the WSDL, 'RunScript' is listed in the 'Methods' section of the "Add Web Reference" dialog.
4. Leave the "Web reference name" text box empty, and click "Add Reference."

Client implementation

The client implementation occurs in the codebehind file. The client class contains a button-click handler that is called when the user clicks `RunScript`. The handler code parses data contained in the form and builds the parameters necessary to make a call to `RunScript`. The client waits for the `RunScript` method to return, then prints the results to the Results text box on the form.

To get a more in-depth look at the actual call to the `RunScript` method, see the codebehind file (`sampleclient-aspnet-vb-soap/Default.aspx.vb` or `sampleclient-aspnet-cs-soap/Default.aspx.cs`).

Troubleshooting

Verify that permissions were set up properly

To ensure that IIS has the permissions needed to display this site, verify that the `IUSR_computer-name` and ASPNET local accounts have read access to each file in the `sampleclient-aspnet-soap` folder in the webroot.

Verify that InDesign Server is listening for SOAP requests

Using a Windows machine, you can verify that InDesign Server is listening on port 12345 by typing `netstat -a` at a command prompt. This command returns a list of ports that are listening. If InDesign Server started successfully, port 12345 is in that list.

“Server Error in '/sampleclient-aspnet-soap/sampleclient-aspnet-cs-soap' Application.

The path '/sampleclient-aspnet-soap/sampleclient-aspnet-cs-soap/App_GlobalResources/' maps to a directory outside this application, which is not supported.”

This error can appear when attempting to display `Default.aspx` in your browser when the IIS local path for the web site is incorrect. To resolve this:

1. Open Internet Information Services (from Start > Control Panel > Administrative Tools).
2. Use the tree view to navigate to the default Web site. Typically this is found at MY-MACHINE (localcomputer) > Web Sites > Default Web Site.
3. Right-click Default Web Site, and select Properties.
4. In the Properties dialog, select the Home Directory tab.
5. In the “Local Path:” text box, remove any trailing backslash (\) characters from the path.
6. Click OK.

“Exception: The URI prefix is not recognized”

This error can appear in the results text after attempting to call `RunScript`. This error indicates that the InDesign Server Host you entered does not have a valid URI prefix such as `http://`; for example (where the port is 12345), `http://localhost:12345` or `http://123.45.678.90:12345`.

“Server Error in '/' Application. Configuration Error”

Description: An error occurred during the processing of a configuration file required to service this request. Please review the specific error details below and modify your configuration file appropriately.

Parser Error Message: It is an error to use a section registered as allowDefinition='MachineToApplication' beyond application level. This error can be caused by a virtual directory not being configured as an application in IIS.

This error can appear when attempting to display `Default.aspx` in your browser when the Web site is not configured as an application in IIS. After creating or copying a folder to IIS's `wwwroot`, you must configure your folder as an application under IIS. To register the folder as an application, follow the steps in [“Install the sample under IIS” on page 12](#).

“Exception: Unable to connect to the remote server”

This error can appear in the results text after attempting to call `RunScript`. This error indicates that either you incorrectly specified the InDesign Server Host URI or InDesign Server is not running at that location.

“Server Error in '/sampleclient-aspnet-soap/sampleclient-aspnet-cs-soap' Application”

This may occur if the ASP.NET version found on the ASP.NET tab under Properties for the Web site or virtual directory in IIS is not set to 2.0.

sampleclient-csharp-soap

This command-line client application is written in C# and communicates with InDesign Server through SOAP. It uses the `RunScript()` method defined in `IDSP.wsdl` to pass a script file to InDesign Server.

The Visual Studio project file, `SampleClient.csproj`, uses a Web reference to `IDSP.wsdl` to generate a Web service proxy class that exposes the Web service methods defined in `IDSP.wsdl`.

The source code comprises two files:

- ▶ `CommandLineArguments.cs` — Handles the parsing of the command-line arguments.
- ▶ `Program.cs` — Contains the `Main()` method and controls the flow of the application

Usage

Syntax	<code>SampleClient -host hostURL scriptPath [-server] [scriptArgs]</code>
Parameters	
<code>-host</code>	The host to target with the script. The targeted instance of InDesign Server.
<code>hostURL</code>	A valid URL (using the prefix <code>http://</code>) that specifies the IP address and port of the server. If the client and server are running on the same machine, this can be <code>http://localhost:portID</code> , where <code>portID</code> is a valid port number.

<i>scriptPath</i>	The full path of the script file to send to the host. The script can be written using JavaScript, VBScript (Windows only), or AppleScript (Mac OS only).
<code>-server</code>	(Optional) Indicates that <i>scriptPath</i> is the name of a script file on the server's file system. If <code>-server</code> is not specified, <i>scriptPath</i> is the name of a script file on the client's file system.
<i>scriptArgs</i>	(Optional) One or more arguments to pass in to the script. Each <i>scriptArg</i> is formatted as <i>name=value</i> . Multiple <i>scriptArgs</i> are separated by a space; for example: arg0=100 arg1=200 arg3="this is some text"

Example `SampleClient -host http://localhost:12345 D:\javaScripts\HelloWorld.jsx -server arg0=100 arg1="this is text"`

Building with Microsoft Visual Studio 2008

1. Open `SampleClient.csproj` in Visual Studio.
2. Choose either the Debug or Release target.
3. Choose Build > Build Solution menu (if necessary, save the `.sln` file to the default location).

If it is necessary to update the `IDSP.wsdl` Web reference, follow these steps:

1. Right-click the `IDSP_WebReference` in the Web references folder in the Solution Explorer, and choose Properties.
2. Choose the InDesign Server WSDL by one of the following two methods:
 - ▷ Choose the "Web Services on the local machine" link, and select `IDSP.wsdl`. (`IDSP.wsdl` is located in `<SDK>/docs/references`).
 - ▷ Enter the WSDL URL in the URL text box (for example, `http://localhost:12345/service?wsdl`), and click Go. This method requests the WSDL at run-time from the specified instance of InDesign Server, so make sure InDesign Server is running at the specified location.
3. Close the properties dialog.
4. Right-click `IDSP_WebReference` again, and choose Update Web Reference. This regenerates the Web service proxy class for `IDSP.wsdl`.

Debugging

1. Right-click the `SampleClient` project in the Solution Explorer, and choose Properties.
2. Click the Debug tab, and enter command-line arguments for `SampleClient`. For example:

`-host http://localhost:12345 D:\javaScripts\HelloWorld.jsx arg0=100 arg1="this is text"`
3. Close the Properties dialog.
4. Set breakpoints in the code as necessary.
5. Choose Debug > Start Debugging.

RunScript results

SampleClient outputs the return value of the script to the console on successful completion of the script. If the return type is an array or an object, SampleClient outputs the return value as an array of XML elements.

sampleclient-flex-soap

This client application is written in Flex3, and it communicates with InDesign Server through SOAP. It uses a Web-service object to call the `runScript()` method provided in the InDesign Server WSDL, to execute a script file within InDesign Server. The script can be JavaScript, AppleScript, or VBScript.

There is one source file, `/src/sampleclient.mxml`.

How to build the sample

There are two options for building the sample: use the FlexBuilder3 project or use the Flex3 SDK command line tool `mxmlc`.

Running Flex Builder 3

1. In the Flex Navigator window, right-click and choose Import...
2. Choose General > Existing Projects into Workspace.
3. Click Next button.
4. Choose "Select root directory," and browse to the `sampleclient-flex-soap` folder.
5. Click Finish.

Using mxmlc

The compiler is located in the `bin` folder of the Flex SDK installation folder. Open a command line or Terminal window, set the current directory to the `bin` folder, then use the following command (change the path as necessary):

```
mxmlc <SDK>/samples/sampleclient-flex-soap/src/sampleclient.mxml
```

This command creates a `bin-debug` folder in the `SDK/samples/sampleclient-flex-soap` folder and copies the compiled SWF file to that location.

Usage

First build the sample, then you can run the SWF from within FlexBuilder or you can open the SWF in a browser. This sample provides you with a user interface for sending a script to run under InDesign Server. Each user-interface element is listed below, along with an explanation of its purpose.

InDesign Server Location

URL is the IP address and port of the InDesign Server instance that you want to run the script. This must be a fully formed URL, and the InDesign Server instance must be running on the specified port when you use the “Call RunScript” button; for example: `http://localhost:12345`.

ScriptArgs

ScriptArgs are used to pass parameters to your script. For more information, including how to use ScriptArgs in your script, see the “Working with InDesign Server SOAP” chapter in *Adobe InDesign Server Solutions*. Click Add to create an entry in the scriptArgs table for each ScriptArg you want to send. Both the name and the value must contain data to be sent to your script.

Script

- ▶ Language — Choose the language of your script.
- ▶ Source — Choose File to send the path of your script to InDesign Server. *Filepath* is the path to a file that is accessible by InDesign Server. Or choose Text, then enter the script text in the text-entry box.

Call RunScript button

Click this button to send the script to the specified instance of InDesign Server. You are taken to the Response tab, where you can see the script result or any error or fault that may have occurred.

Troubleshooting

“A Fault occurred while attempting to run RunScript.

fault code: Server.Error.Request

fault string: HTTP request error

fault detail: Unable to load WSDL. If currently online, please verify the URI and/or format of the WSDL (`http://localhost:12345/service?wsdl`)”

This fault occurs if the specified instance of InDesign Server is not accessible. Make sure the URL is fully formed (including the `http://` prefix) and an instance of InDesign Server is running on the specified machine and port.

This error also may occur if the Flash player has insufficient permissions for accessing network resources. See [“Problems communicating from your SWF to InDesign Server” on page 20](#).

“A Fault occurred while attempting to run RunScript.

fault code: Server.Error.Request

fault string: HTTP request error

fault detail: Error: [IOErrorEvent type='ioError' bubbles=false

cancelable=false eventPhase=2 text='Error #2032: Stream Error. URL: http://localhost:12345']. URL: http://localhost:12345'

This fault occurs if the specified script file does not exist.

Problems communicating from your SWF to InDesign Server

If you have problems communicating from your SWF to an instance of InDesign Server on another machine, you need to give your SWF permission to access the other machine. Set your SWF's sandbox type to `localTrusted`. To do this, modify your FlexBuilder project's compile arguments:

1. Open the Properties dialog for your project.
2. Go to the Flex Compiler panel.
3. Add the following to the compiler arguments: `-use-network`. This sets the `use-network` setting to `false`.

In addition, you may need to grant additional security permissions for your SWF. Do one of the following:

- ▶ Use the Flash Global Security Settings Panel to add your InDesign Server machine to the list of trusted URLs. The settings panel is accessed from:
http://www.macromedia.com/support/documentation/en/flashplayer/help/settings_manager04.html#117502

— OR —

- ▶ Add a configuration file to the `FlashPlayerTrust` folder on the local machine, as described here:
 - ▷ Locate the `FlashPlayerTrust` folder on your local computer.
 - ▷ Add a new text file to the `FlashPlayerTrust` folder with the location of the SWF folder to be granted additional permissions.
 - ▷ Save that text file with a unique name and a `.CFG` extension.

See http://livedocs.adobe.com/flex/3/html/help.html?content=security2_25.html.

sampleclient-php-soap

The `sampleclient-php-soap` sample is a PHP script that uses SOAP to communicate with InDesign Server. This sample provides client code for two major PHP SOAP implementations: PHP:SOAP and NuSOAP.

Requirements

PHP is a server-side scripting language that can be embedded directly into HTML code. PHP often is used together with Apache Web Server (<http://httpd.apache.org/>), but it also can be used with Microsoft's Internet Information Services (IIS) (<http://www.iis.net/>) on Windows. We assume you have a PHP-compatible Web server installed before attempting to use the sample.

We also assume you have your Web server configured to use PHP. For help configuring your Web server, see one of these guides:

- ▶ <http://us2.php.net/manual/en/install.php>

- ▶ <http://www.thesitewizard.com/archive/php4install.shtml>
- ▶ <http://developer.apple.com/internet/opensource/php.html>

You also must install the desired PHP SOAP implementation(s). This sample was developed using the following:

- ▶ PHP 5.2.1
- ▶ PHP:SOAP 5.2.1
- ▶ NuSOAP 0.7.2

For downloads, information, and tutorials, visit these sites:

- ▶ <http://sourceforge.net/projects/nusoap/>
- ▶ <http://www.scottnichol.com/nusoapintro.htm>
- ▶ <http://phpsoaptoolkit.sourceforge.net/phpsoap/>
- ▶ <http://www.w3schools.com/php/>

Required modifications to NuSOAP source code

Both NuSOAP and PHP:SOAP use the class name `soapclient` within their API. To avoid a name clash, you must modify `nusoap.php` to change all instances of `soapclient` to `nusoapclient`. If you have not installed PHP:SOAP and do not want to modify the NuSOAP code, you must modify `sampleclient_php_nusoap.php` to change all instances of `nusoapclient` to `soapclient`.

Install the sample under Apache or IIS

Copy the sample folder, `sampleclient-php-soap`, from the InDesign Server SDK to the appropriate location within your Web server's default Web site folder. Typically, the Apache default Web site folder is `C:\Program Files\Apache Software Foundation\Apache2.2\htdocs\`. The IIS default Web site folder typically is `C:\Inetpub\wwwroot`.

Using the sampleclient

1. Using a Web browser, load the `sampleclient_php_main.php` page. If the sample's folder is located at the root of your Web server, the address of the script is:

`http://localhost/sampleclient-php-soap/sampleclient_php_soap_main.php`
2. Launch InDesign Server for use with SOAP (for example, `indesignserver -port 12345`).
3. Fill in the information on the client's Web form:
 - ▷ **InDesign Server Host** — Enter the URI of the desired InDesign Server instance (for example, `http://localhost:12345`).
 - ▷ **Choose a script** — Select `MakeDocument.jsx`. The listed scripts are read from the `sampleclient-php-soap/IDS_Scripts` folder. Files added to the folder are automatically displayed in the list. AppleScript scripts will run only on a Mac OS version of InDesign Server, and

VBScripts will run only on a Windows version of InDesign Server. Error #30485 will occur if you choose an incompatible script.

- ▷ Create a scriptArg with the name `saveAs` and the value set to a valid path for an InDesign document (for example, `c:\myNewFile.indd`). The path must be based on the file system of the targeted instance of InDesign Server.
- ▷ Click Run for your desired client.

The script is sent to and run by the targeted instance of InDesign Server. The script creates a new, empty InDesign document and saves it to the path specified by the `saveAs` scriptArg. The main page then reloads and displays the resulting path of the created file. If the `$doDebug` variable in `sampleclient_php_soap_main.php` is true, you also will see the XML request and response packets.

Structure of sampleclient-php-soap

The sample comprises the following files:

- ▶ `sampleclient_php_soap_main.php` — This is the main script. It presents the user with an HTML form used to enter data necessary to run one of the clients. It presents the user with two buttons, one for each client. When a button is clicked, the main page is reloaded, and the selected client is loaded by including the appropriate PHP file. It then parses the form data and calls the client's `phpRunScript` method.
- ▶ `sampleclient_php_nusoap.php` — This script uses NuSOAP to call the InDesign Server `RunScript` method via SOAP.
- ▶ `sampleclient_php_phpsoap.php` — This script uses PHP:SOAP to call the InDesign Server `RunScript` method via SOAP.

Troubleshooting

For PHP:SOAP, you must correctly set the `extension_dir` in `php.ini`. The default value is `.\ext`, which may not be appropriate for your installation. Setting the directory to the full path may work (for example, `c:\Program Files\php\ext`).

If you get the following error with PHP5 and NuSOAP—"Cannot redeclare class soapclient"—it is because both PHP:SOAP and NuSOAP declare a class named `soapclient`. Rename all occurrences of `soapclient` to `nusoapclient` in all your NuSOAP files. Alternately, disable the `php_soap` extension in your `php.ini` file. Back up your files before making either of these changes.

3 Java/CORBA Samples

This chapter contains information about the Java/CORBA samples contained in the Adobe® InDesign® Server SDK. These samples demonstrate how to interact with InDesign Server using Java/CORBA.

Introduction

The Java samples demonstrate the use of the InDesign Server Java API, which uses CORBA as its communication protocol. For more information on the Java API, see “Working With Adobe InDesign Server Java” in *Adobe InDesign Server Solutions*.

- ▶ [idsp-wsdl-java](#) — This is a helper project that generates Java code for the InDesign Server WSDL file, `IDSP.WSDL`.
- ▶ [sampleclient-java-corba](#) — This sample uses `doScript` to run a script under InDesign Server.
- ▶ [snippets](#) — This sample contains almost 200 Java snippets demonstrating how to use the InDesign Server Java API. For a list of the snippets, see `/samples/snippets/ids-sdk-snippets.htm`.

idsp-wsdl-java

This project is used to generate java code based on the InDesign Server WSDL file, `IDSP.WSDL`. It uses the Axis tool `wsdl2java` to generate the code. The Axis tool is located within `axis.jar` in the `<SDK>/external/axis-1_4/lib` folder. The Ant file, `build.xml`, contains the commands to execute the `wsdl2java` tool, compile the generated source code, and build a JAR file.

Building with Ant

Open a shell window, and set the current directory to the project folder, where `build.xml` is located:

```
cd <SDK>/samples/idsp-wsdl-java
```

Then execute the Ant build script:

```
ant
```

The Ant build script compiles the source code, and outputs a JAR file to the `lib` folder.

Using Eclipse

An Eclipse project file is provided in the project folder. To open the `idsp-wsdl-java` Eclipse project file, run the Eclipse IDE, and follow these steps:

1. Import the project:
 - ▶ From the main menu bar, choose `File > Import...` to open the Import wizard.
 - ▶ Choose `General > Existing Projects into Workspace`, and click `Next`.
 - ▶ Choose “Select root directory:”, and click `Browse...`

- ▷ In the file dialog, select the root folder of the SDK, and click OK.
 - ▷ A list of available SDK projects appears in the Projects window. The `idsp-wsdl-java` project depends on the `external` project, so make sure to check that project as well as the `idsp-wsdl-java` project, then click Finish.
- If the desired projects do not show up in the Projects window, they probably already exist in the workspace.
2. Use Ant to build the project and JAR file: right-click on the project's `build.xml` file in the Package Explorer window, and choose Run As > Ant Build. This builds the source code and exports the JAR file.

Build warnings

“Unable to find required classes (javax.activation.DataHandler and javax.mail.internet.MimeMultipart). Attachment support is disabled.”

This warning does not affect the ability to run `sampleclient-java-soap`, as attachment support is not a required element. To remove the warning, you need to install the Java Mail Framework and JavaBeans Activation Framework, then add `mail.jar` and `activation.jar` to your classpath.

`mail.jar` is supplied in the Java Mail Framework. It can be downloaded from:

<http://java.sun.com/products/javamail/downloads/index.html>

`activation.jar` is supplied in the JavaBeans Activation Framework. It can be downloaded from:

<http://java.sun.com/javase/technologies/desktop/javabeans/jaf/>

Java language warnings

The InDesign CS5 Server SDK supports Java 1.4.2, so it does not make use of Java elements newer than 1.4.2. If your Java version is newer, you may experience build warnings like the following:

- ▶ “References to generic type `Vector<E>` should be parameterized”
- ▶ “The serializable class `class_name` does not declare a static final `serialVersionUID` field of type `long`”

These warnings can be safely ignored.

sampleclient-java-corba

This client application is written in Java, and it communicates with InDesign Server through CORBA. It uses the `doScript()` method provided in the InDesign Server Java API to execute a JavaScript file within InDesign Server.

The source code comprises three files:

- ▶ `CommandLineArguments.java` — Handles the parsing of the command line arguments.
- ▶ `SampleClient.java` — Contains the `main()` method, and controls the flow of the application.
- ▶ `VariableTypeOutputUtils.java` — Prints variable type data to a print stream.

Usage

Syntax `SampleClient -iorfile filepath scriptpath [-server] [-repeat number] {scriptArgs}`

Parameters

<code>-iorfile filepath</code>	<code>filepath</code> is the path of the OmniORB IOR file written by the targeted server at startup.
<code>scriptpath</code>	The path of the JavaScript file to execute.
<code>-server</code>	Indicates that <code>scriptpath</code> is the name of a script file on the server's file system. If <code>-server</code> is not specified, <code>scriptpath</code> is the name of a script file on the client's file system.
<code>-repeat number</code>	Execute the script <code>number</code> of times.
<code>scriptArgs</code>	The arguments to pass into the script. Each <code>scriptArg</code> is formatted as "name=value", and multiple <code>scriptArgs</code> are separated by spaces. For example: <code>arg0=100 arg1=200 arg3="this is some text"</code>

NOTE: When launching the sample client, InDesign Server must first be running and configured for use with CORBA using the designated IOR file. To learn how to launch InDesign Server, see *Introduction to Adobe InDesign CS5 Server Development*.

Running from the command line

For your convenience, scripts are provided to execute the sample client. The scripts are located in the project's `scripts` folder. To execute the script, first open a shell window (Command Prompt on Windows, or Terminal on Mac OS) and set the current directory to the project's `scripts` folder, then execute the script with the appropriate parameters. For example:

Windows

```
\cd "c:\Program Files\Adobe InDesign CS5 Server SDK\
  samples\sampleclient-java-corba\scripts"
sampleclient -iorfile c:\ior.txt "c:\myJavaScripts\myscript.jsx" arg0=1 arg1="hello"
```

Mac OS

```
cd "/Applications/Adobe InDesign CS5 Server SDK/
  samples/sampleclient-java-corba/scripts"
sh sampleclient.sh -iorfile /ior.txt "/myJavaScripts/myscript.jsx" arg0=1 arg1="hello"
```

Output is written to the shell window and/or the InDesign Server console.

Building with Ant

Open a shell window, and set the current directory to the project folder, where `build.xml` is located:

```
cd <SDK>/samples/sampleclient-java-corba
```

Then execute the Ant build script:

```
ant
```

The Ant build script compiles the source code and outputs a JAR file to the `lib` folder.

Using Eclipse

An Eclipse project file is provided in the project folder. To open the `sampleclient-java-corba` Eclipse project file, run the Eclipse IDE, and follow these steps:

1. Import the project:
 - ▷ From the main menu, choose File > Import... to open the Import wizard.
 - ▷ Choose General > Existing Projects into Workspace, and click Next.
 - ▷ Choose "Select root directory:" and click Browse...
 - ▷ In the file dialog, select the root folder of the SDK, and click OK.
 - ▷ A list of available SDK projects appears in the Projects window. The `sampleclient-java-corba` project depends on the `InDesignServerAPI` project, so make sure to check that project as well as the `sampleclient-java-corba` project, then click Finish.

If the desired projects do not show up in the Projects window, they probably already exist in the workspace.
2. Use Ant to build the project and JAR file: right-click on the project's `build.xml` file in the Package Explorer window, and choose Run As > Ant Build. This builds the source code and exports the JAR file.

Running from within Eclipse

1. In the Project Explorer, select the root folder of `sampleclient-java-corba`.
2. From the main menu, choose Run > Run... to open the Run wizard.
3. Click "New launch configuration."
4. In the Name box, enter a name for the launch configuration.
5. Select the Arguments tab, and enter the arguments necessary to run; for example:

```
-iorfile c:\ior.txt "c:\MyJavaScripts\myscript.jsx" arg0=1 arg1="hello"
```
6. Click Run.

Once you create the run configuration, you need to just select it and click Run the next time.

Output is written to the Eclipse console and/or the InDesign Server console.

Java language warnings

The InDesign CS5 Server SDK supports Java 1.4.2, so it does not make use of Java elements newer than 1.4.2. If your Java version is newer, you may experience build warnings like the following:

- ▶ "References to generic type Vector<E> should be parameterized"
- ▶ "The serializable class class_name does not declare a static final serialVersionUID field of type long"

These warnings can be safely ignored.

snippets

This project contains many Java code snippets which communicate with InDesign Server through CORBA, using the InDesign Server Java API. The intent of this project is to provide you with sample code that teaches you how to interact with InDesign Server to extend its functionality, enabling you to create specialized workflows.

The source code is organized into groups designated by the area of InDesign Server the snippet uses. Each snippet contains a `main()` method, so it can be executed individually.

To learn more about the snippets, see `ids-sdk-snippets.htm` in the `snippets` folder. This file contains the name, package, description, and arguments for each snippet. While the HTML file is helpful, most snippets are written with the expectation that you will be reading the code to understand what it does and know what input/output it requires.

Example XML data files are supplied in the `examplefiles` folder. Some snippets look for the example files in `C:\examplefiles` by default, so if you are on a Windows platform, you can copy the `examplefiles` folder to your C drive; otherwise, you can override the default file location by passing in the path to the example file as an argument to the snippet as necessary.

Usage

Syntax `runsnippet partialpackage.classname {args}`

Parameters `partialpackage` The remainder of the package name that comes after `com.adobe.ids.sdk`. (for example, `document` or `application`). This is case sensitive.

`classname` The name of the class to execute. This is case sensitive.

`args` The arguments to pass into the component. Typically, the first argument is the path to the IOR file. Arguments are formatted as individual strings, and they are separated by spaces.

Example `c:\ior.txt 100 "this is some text"`

NOTE: Before running a snippet, InDesign Server must be running and configured for use with CORBA using the designated IOR file. To learn how to launch InDesign Server, see *Introduction to Adobe InDesign CS5 Server Development*.

Running from the command line

For your convenience, scripts are provided to execute the snippets. The scripts are located in the project's `scripts` folder. To execute the script, first open a shell window (Command prompt on Windows or Terminal on Mac OS) and set the current directory to the project's `scripts` folder, then execute the script with the appropriate parameters. For example:

Windows

```
cd "c:\Program Files\Adobe InDesign CS5 Server SDK\samples\snippets\scripts"
runsnippet HelloWorld c:\ior.txt
runsnippet document.OpenDocument c:\ior.txt "c:\my documents\myDesign.indd"
```

Mac OS

```
cd "/Applications/Adobe InDesign CS5 Server SDK/samples/snippets/scripts"
sh runsnippet.sh HelloWorld /ior.txt
sh runsnippet.sh document.OpenDocument /ior.txt "/my documents/myDesign.indd"
```

Output is written to the shell window and/or the InDesign Server console. Also, if an InDesign file is generated by the snippet, the file is output to the `output` folder inside the scripts folder.

Building with Ant

1. Open a shell window, and set the current directory to the project folder, where `build.xml` is located:

```
cd SDK/samples/snippets
```

2. Execute the Ant build script:

```
ant
```

The Ant build script compiles the source code and outputs a JAR file to the `lib` folder.

Using Eclipse

An Eclipse project file is provided in the project folder. To open the snippets Eclipse project file, run the Eclipse IDE, and perform the following steps:

1. Import the project:

- ▷ From the main menu, choose File > Import... to open the Import wizard.

- ▷ Choose General > Existing Projects into Workspace, and click Next.

- ▷ Choose "Select root directory:" and click Browse...

- ▷ In the file dialog, select the root folder of the SDK, and click OK.

- ▷ A list of available SDK projects appears in the Projects window. The `snippets` project depends on the `InDesignServerAPI` project, so make sure to check both projects, then click Finish.

If the required projects do not show up in the Projects window, they probably already exist in the workspace.

2. Use Ant to build the project and JAR file: right-click on the project's `build.xml` file in the Package Explorer window, and choose Run As > Ant Build. This builds the source code and exports the JAR file.

Running from within Eclipse

1. In the Project Explorer, select the snippet you want to run.

2. From the main menu, choose Run > Run... to open the Run wizard.

3. Click "New launch configuration."

4. In the Name box, enter a name for the launch configuration.

5. Select the Arguments tab, and enter the arguments necessary to run; for example:

```
c:\ior.txt "c:\my documents\myDesign.indd"
```

6. Click Run.

After you create the run configuration, you need to just select it, enter proper arguments for the snippet, and click Run the next time.

Output is written to the Eclipse console and/or the InDesign Server console. Also, if an InDesign file is generated by the snippet, the file is output to the `output` folder inside the Snippets folder.

Java language warnings

The InDesign CS5 Server SDK supports Java 1.4.2, so it does not make use of Java elements newer than 1.4.2. If your Java version is newer, you may experience build warnings like the following:

- ▶ “References to generic type `Vector<E>` should be parameterized”
- ▶ “The serializable class `class_name` does not declare a static final `serialVersionUID` field of type `long`”

These warnings can be safely ignored.

snippets-runner

This project builds a tool that is used in conjunction with the snippets sample. A snippet is defined as a small Java application that uses the InDesign Server Java API. `SnippetRunner` is a Java application that opens the `snippets.jar` file, and builds a list of the snippets therein. Only classes that contain a `Main()` method are chosen. This list of snippets is then displayed in the `SnippetRunner` user interface, where you can choose a snippet, set up a parameter list, and run the snippet.

The source code for `SnippetRunner` is in the `snippets-runner` folder.

Quick start

1. Use Ant to build the `snippets` sample:

```
cd <SDK>/samples/snippets
ant
```

2. Use Ant to build the `snippets-runner` sample:

```
cd <SDK>/samples/snippets-runner
ant
```

3. Run InDesign Server for use with CORBA:

```
cd <IDS>
indesignserver -iorfile /c/ior.txt -pluginpath server/corba
```

4. Run `SnippetRunner` (in a new shell window):

```
cd <SDK>\samples\snippets-runner          Windows
SnippetRunner
```

— OR —

```
cd <SDK>/samples/snippets-runner          MAC OS
sh snippetrunner.sh
```

Usage

Snippet runner has a full GUI. Its elements are described below:

- ▶ **Snippet Jar** — This is the `snippets.jar` file that SnippetRunner uses to execute snippets. By default, this is `samples/snippets/lib/snippets.jar`. To change which file you use, click “Choose snippets.jar.” You must first build the `snippets.jar` file.
- ▶ **IOR file** — This is the IOR file used to determine the instance of InDesign Server under which you will run the snippet. To change the IOR file setting, click “Choose IOR file.”
- ▶ **Choose a Snippet** — This combo-box contains all available snippets. Choose which snippet you want to run from the pop-up menu.
- ▶ **Enter parameter list** — Here, you enter the parameter list to be sent to the snippet. All snippets require the path to the IOR file as their first parameter. This parameter is supplied for you; do not enter it in the text-entry box. For snippets that require additional parameters, you will see the names of those parameters listed above the text entry. Parameters are separated by spaces. Use quotes around parameters that contain a space. As an example, try selecting the `application.DoRunScript` snippet.

For example:

```
"c:\my folder\myFile.indd" 100 "more text"
```

- ▶ **Description** — This describes the functionality of and additional parameters to the snippet.
- ▶ **Output** — `stdout` and `stderr` are rerouted to this text area. `Consoleout` calls in the snippet result in output to the InDesign Server Console window. To delete the text in the box, click “Clear Output.”

Running from the command line

For your convenience, scripts have been provided to run SnippetRunner. The scripts are located in the `snippets-runner` folder. To execute the script, first open a shell window (Terminal on Mac, Command Prompt on Windows) and set the current directory to the `snippets-runner` folder, then execute the script:

Windows `cd <SDK>\samples\snippets-runner`
`SnippetRunner`

Mac OS `cd <SDK>/samples/snippets-runner`
`sh snippetrunner.sh`

Building with Ant

1. Open a shell window, and set the current directory to the project folder, where `build.xml` is located:

```
cd <SDK>/samples/snippets-runner
```

2. Execute the ant build script:

```
ant
```

The Ant build script compiles the source code and outputs a JAR file to the `lib` folder.

Using Eclipse

An Eclipse project file is provided in the project folder. To open the snippets-runner Eclipse project file, run the Eclipse IDE, and perform the following steps:

1. Import the project:

- ▷ From the main menu bar, choose File > Import... to open the Import wizard.
- ▷ Choose General > Existing Projects into Workspace, and click Next.
- ▷ Choose "Select root directory;" and click the associated Browse... button.
- ▷ In the file dialog, select the root folder of the SDK, and click OK.
- ▷ A list of available SDK projects appears in the Projects window. The `snippets-runner` project depends on the `InDesignServerAPI` project, so make sure to check both projects, then click Finish.

If the required projects do not show up in the Projects window, they probably already exist in the workspace.

2. Use Ant to build the project and JAR file: right-click on the project's `build.xml` file in the Package Explorer window, and choose Run As > Ant Build. This builds the source code and exports the JAR file.

Running from within Eclipse

1. In the Project Explorer, select the project.
2. From the main menu bar, choose Run > Run... to open the Run wizard.
3. Click "New launch configuration."
4. In the Name box, enter a name for the launch configuration.
5. Set the main class to `com.adobe.ids.sdk.SnippetRunner`.
6. Click Run.

After you create the run configuration, you need to just select it and click Run the next time.

4 COM Samples

This chapter contains information about the COM samples contained in the Adobe® InDesign® Server SDK. These samples demonstrate how to interact with InDesign Server using COM.

Introduction

The COM samples demonstrate the use of the InDesign Server COM type library. For more information about using COM with InDesign Server, see *Introduction to Adobe InDesign CS5 Server Development*.

- ▶ VB: [helloworld-vb-com](#) — This sample create a `helloworld.indd` file using the InDesign Server COM written in Visual Basic.
- ▶ C#: [helloworld-csharp-com](#) — This sample creates a `helloworld.indd` file using the InDesign Server COM written in C#.
- ▶ C#: [sampleclient-csharp-com](#) — This sample sends a script to InDesign Server using the `doScript` method in the InDesign Server COM.

helloworld-vb-com

This simple command-line application is written in Visual Basic and communicates with InDesign Server through COM. It creates a “Hello World” InDesign document and saves the document to the hard drive.

The source code comprises one file, `Module1.vb`. It contains the `Main()` method and controls the flow of the application.

To create a Visual Basic InDesign Server COM component, we recommend that you use Visual Basic 6. Newer versions of Visual Basic have introduced strongly typed constructs that make it less compatible with the loosely typed InDesign Server DOM.

About InDesign Server COM and Visual Basic

It is possible to use Visual Basic .NET, as long as you set the project’s compile options “Option explicit” and “Option strict” to Off. These settings allow you to reduce the strong typing rules of VB.NET.

Avoid declaring variables using `Dim`, as that forces the type onto the variable. Because code completion works only if a variable is declared using `Dim`, you typically can declare your variable at the top of the method using `Dim`, then `REM` the `Dim` statements out before running your code.

Usage

Syntax This application is called from the command line as follows:

```
helloworld-vb-com output_folder config_name
```

Parameters *output_folder* The path to the folder where you want to save the resulting `helloworld` INDD files. The folder should already exist on your system. It is not created for you when the command is run.

config_name The configuration(s) to target with the script. You can specify one or more configurations, separated by a space. Each *config* must be the configuration name of an active InDesign Server instance. When InDesign Server starts up, it creates a configuration name and stores it in the ROT table. This name also is used to create a folder on the hard drive. The following examples demonstrate how the configuration name is created:

InDesign Server Start-up Command Line	Configuration Name
<code>indesignserver -port 12345</code>	<code>configuration_12345</code>
<code>indesignserver -configuration "config1"</code>	<code>config1</code>
<code>indesignserver</code>	<code>configuration_noport</code>

Example If InDesign Server is started with the command `indesignserver -port 12345`, use the following command to target that instance of InDesign Server:

```
helloworld-vb-com C:\ServerTestFiles configuration_12345
```

Multiple configurations can be specified by separating them with a space:

```
helloworld-vb-com C:\ServerTestFiles configuration_12345 config1
```

Building with Microsoft Visual Studio 2008

1. Open `helloworld-vb-com` in Visual Studio.
2. Choose either the Debug or Release target.
3. Choose Build > Build Solution. (If necessary, save the `.sln` file to the default location.)

The executable is created in the `bin\Debug` or `bin\Release` folder.

You probably will need to update the COM reference used in the project file:

1. In the Solution Explorer, right-click the `helloworld-vb-com` project and choose Properties.
2. Go to the References tab.
3. Choose the existing InDesign Server reference, and click Remove.
4. Click Add, and choose Reference...
5. In the Add Reference dialog, go to the COM tab.
6. Choose Adobe InDesign Server CS5 Type Library.
7. Click OK

Debugging

1. Set breakpoints in the code as necessary.
2. Choose Debug > Start Debugging.

Output

`helloworld-vb-com` outputs an INDD file containing a text frame with `helloworld` text. The file(s) are saved to the output folder specified by the `output_folder` parameter.

helloworld-csharp-com

This simple command-line application is written in C# and communicates with InDesign Server through COM. It creates a Hello World InDesign document and saves the document to the specified folder.

Although this sample is here to demonstrate how to build very simple C# COM components, we do not recommend using C# to build InDesign Server COM components. C# is a strongly typed language and has many conflicts with the loosely typed InDesign Server scripting DOM. This `helloworld` sample was possible to write because we used careful type-casting and parameter specification. If your solution requires C#, it is preferable to have your C# code call the `doScript` method of InDesign Server to run a script, as you can see in the `sampleclient-csharp-com` sample.

The source code comprises one file, `Program.cs`. It contains the `Main()` method and controls the flow of the application.

Usage

Syntax This application is called from the command line as follows:

```
helloworld-csharp-com output_folder config_name
```

Parameters `output_folder` The path to the folder where you want to save the resulting `helloworld` INDD files.

`config_name` The configuration(s) to target with the script. You can specify one or more configurations, separated by a space. Each `config` must be the configuration name of an active InDesign Server instance. When InDesign Server starts up, it creates a configuration name and stores it in the ROT table. This name also is used to create a folder on the hard drive. The following examples demonstrate how the configuration name is created:

InDesign Server Start-up Command Line	Configuration Name
<code>indesignserver -port 12345</code>	<code>configuration_12345</code>
<code>indesignserver -configuration "config1"</code>	<code>config1</code>
<code>indesignserver</code>	<code>configuration_noport</code>

Example `helloworld-csharp-com C:\ServerTestFiles configuration_12345 config1`

Building with Microsoft Visual Studio 2008

1. Open `helloworld-csharp-com.csproj` in Visual Studio.

2. Choose either the Debug or Release target.
3. Choose Build > Build Solution. (If necessary, save the .sln file to the default location.)

The executable is created in the `bin\Debug` or `bin\Release` folder.

You probably will need to update the COM reference used in the project file:

1. In the Solution Explorer, open the References folder.
2. Right-click the existing InDesign Server reference, and choose Remove.
3. Right-click the References folder, and choose Add Reference...
4. In the Add Reference dialog, go to the COM tab.
5. Choose Adobe InDesign Server CS5 Type Library.
6. Click OK.

sampleclient-csharp-com

This command-line client application is written in C# and communicates with InDesign Server through COM. It uses the `doScript()` method defined on the InDesign Server Application object to run a script under InDesign Server. The script can be written in JavaScript or VB Script. There is a sample script included in the `samplescript` folder. The script simply returns the values of the parameters sent to the script.

The source code comprises two files:

- ▶ `CommandLineArguments.cs` — Parses command-line arguments.
- ▶ `Program.cs` — Contains the `Main()` method and controls the flow of the application.

Usage

Syntax This application is called from the command line as follows:

```
sampleclient-csharp-com -config config_name scriptpath [ -server ] [ "scriptParams" ]
```

Parameters *config_name* The configuration(s) to target with the script. You can specify one or more configurations, separated by a space. Each *config* must be the configuration name of an active InDesign Server instance. When InDesign Server starts up, it creates a configuration name and stores it in the ROT table. This name also is used to create a folder on the hard drive. The following examples demonstrate how the configuration name is created:

InDesign Server Start-up Command Line

```
indesignserver -port 12345
indesignserver -configuration "config1"
indesignserver
```

Configuration Name

```
configuration_1234
config1
configuration_noport
```

scriptpath The filepath for the script to be sent to InDesign Server.

<code>-server</code>	Indicates that <code>scriptpath</code> is the name of a script file on the server's file system. If <code>-server</code> is not specified, <code>scriptpath</code> is the name of a script file on the client's file system.
<code>scriptParams</code>	The parameters to pass in to the script. These are ordered parameter values, separated by spaces. Strings with spaces must be surrounded with quotes (for example, "hello there" 100).

Example

```
sampleclient-csharp-com -config configuration_12345  
..\..\samplescript\doScriptArgs.jsx "hello there" 100
```

Building with Microsoft Visual Studio 2008

1. Open `sampleclient-csharp-com.csproj` in Visual Studio.
2. Choose either the Debug or Release target.
3. Choose Build > Build Solution. (If necessary, save the `.sln` file to the default location.)

The executable is created in the `bin\Debug` or `bin\Release` folder.

You probably will need to update the COM reference used in the project file:

1. In the Solution explorer, open the References folder.
2. Right-click the existing InDesign Server reference, and choose Remove.
3. Right-click the References folder, and choose Add Reference...
4. In the Add Reference dialog, go to the COM tab.
5. Choose Adobe InDesign Server CS5 Type Library.
6. Click OK.

Debugging

1. In the Solution Explorer, right-click the `sampleclient-csharp-com` project, and choose Properties.
2. Click the Debug tab, and enter command-line arguments for `sampleclient-csharp-com`; for example:

```
-config configuration_12345 "..\..\samplescript\doScriptArgs.jsx" "hello" 100
```

3. Close the Properties dialog.
4. Set breakpoints in the code as necessary.
5. Choose Debug > Start Debugging.

Script results

`sampleclient-csharp-com` outputs the return value of the script to the console on successful completion of the script. If the return type is an array or an object, `sampleclient-csharp-com` outputs the return value as an array of values.