ADOBE® INDESIGN® CS5 SERVER

INTRODUCTION TO ADOBE INDESIGN CS5 SERVER

Adobe

Introduction to Adobe InDesign CS5 Server

Technical note #10123

Contents

Introduction to Adobe InDesign CS5 Server

This document is an overview of Adobe® InDesign® Server. It describes how to install and run InDesign Server in a simple environment and briefly describes how to communicate with InDesign Server from external components.

InDesign Server is used by systems integrators and solution developers to build server-based publishing solutions. You can think of InDesign Server as a "headless" (that is, no user interface) version of InDesign. It is built on the same code base as the desktop version of InDesign, but it was adapted for server use. The adaptations include support for multiple instances, error capturing and logging, and communication through SOAP and CORBA.

Like its desktop counterpart, InDesign Server offers a rich development environment. It supports open standards like XML, and it is highly scriptable and extensible. InDesign Server can be extended by using the APIs supplied in the InDesign products SDK and the InDesign Server SDK. These APIs can be used to create C++ plug-ins, Java™ components, JavaScript scripts, AppleScript scripts, and VBScript scripts.

Terminology

This section defines terms used throughout this document:

- *Ant* — Another Neat Tool, a Java-based build tool.
- *Client* — A requester of services from a server in a distributed computing system.
- *CORBA* — Common Object Request Broker Architecture, a language-independent, distributed object model.
- COM — Component Object Model, a technology facilitating interoperability between applications and components.
- *IDS* — InDesign Server.
- *IOR file* — Interoperable Object Reference file. This is used by *CORBA* and its clients to identify an object in the CORBA object model.
- *IPv6* — Internet Protocol Version 6, the successor to IPv4. IPv6 is supported by InDesign CS4 Server and later.
- *Job queueing* — A mechanism that queues submitted jobs, executes them asynchronously, and provides a means to later check a jobs status.
- *Load balancing* — Distributing jobs across multiple instances and servers based on availability.
- *LBQ* - An add-on component introduced with InDesign CS5 Server that performs job queueing and load balancing for multiple IDS instances.
- *ORB* — Object Request Broker, a piece of software that implements the *CORBA* object model.

- *<SDK>* — The path where you installed the InDesign Server SDK.
- *Server* — The provider of services in a distributed computing system.
- *Shell window* — A command-line window. On Windows®, use Command Prompt (located in the Accessories folder from the Start menu). On Mac OS®, use the Terminal utility (located in /Applications/Utilities).
- *SOAP* — Simple Object Access Protocol, an XML-based protocol for exchanging messages between programs and platforms. InDesign Server supports several versions of SOAP: SOAP industry standard Versions 1.1 and 1.2 (RPC and doc/lit) and *WSDL* 1.1.
- *WSDL* — Web Service Description Language, an XML-based format for describing how to access a Web service and what operations it will perform.

Roadmap

The following sections describe how to work with and extend InDesign Server.

Documentation

Both InDesign Server SDK and InDesign Products SDK contain documentation and sample code to teach you how to extend InDesign Server. Both SDKs are available for download from Adobe.com. The document titles mentioned within this document are located in the SDKs.

Scripting

Like InDesign, InDesign Server provides scripting support for JavaScript, AppleScript, and VBScript.

Write a script

To learn how to write InDesign Server scripts, read the following documents:

- *Adobe InDesign CS5 Scripting Tutorial*
- *Adobe InDesign CS5 Scripting Guide*
- *Adobe InDesign CS5 Server Scripting Guide*
- Also, you can access the InDesign CS5 Object Model Viewer from within the Help menu of Adobe ExtendScript Toolkit CS5.

Run a script

InDesign Server provides one SOAP method, RunScript, that is used to send a script to InDesign Server. Your installation of InDesign Server provides a simple SOAP client, *SampleClient*, that you can use to run a script. See "Using sampleclient to run an InDesign Server script" on page 21.

SampleClient was written using C++, and its source code and project files are provided in the InDesign products SDK. The InDesign Server SDK contains other sample client projects written in several languages, including Java, C#, PHP, VB.NET, and Flex.

Java

Write a Java Component

InDesign Server provides a Java API that communicates with InDesign Server through CORBA. For details, see "Interfacing with InDesign Server through Java" on page 24 and *Working with Adobe InDesign CS5 Server Java*.

C++

Write a C++ Plug-in

InDesign Server is written in C++ and provides support for C++ plug-ins developed with the InDesign C++ API. For details, refer to "Interfacing with InDesign Server through a plug-in" on page 25, *Adobe InDesign CS5 Programming Guide*, and *Adobe InDesign CS5 Server Plug-in Techniques*.

Access your Plug-in using scripting or Java

To access your plug-in from a script or from Java, you must make your plug-in scriptable. For details, refer to the "Scriptable Plug-in Fundamentals" chapter in the *Adobe InDesign CS5 Programming Guide*.

To access your plug-in from Java, you also need to regenerate the Java API JAR file (InDesignServerAPI.jar) and CORBA Support plug-in, as described in *Regenerating the Adobe InDesign CS5 Server Java API*.

COM

InDesign Server publishes a COM type library that you can use to write COM components that interoperate with InDesign Server. COM components can be written in several languages. This document briefly discusses using Visual Basic and C# in "Interfacing with InDesign Server through COM (Windows only)" on page 26.

Windows Service

You can use a Windows Service to manage InDesign Server. To employ the service, first install InDesignServerService as described in "Installing the InDesign Server Windows Service (Windows only)" on page 11. Then you can use the InDesign Server Microsoft Management Console snap-in to configure InDesignServerService.

launchd Daemon

You can use a launchd daemon to manage InDesign Server on Mac OS. To learn how to configure a daemon, see "Configuring an InDesign Server launchd daemon for Mac OS" on page 19.

Performance and scalability

Documentation and tools related to the performance and scalability of InDesign Server are available in the InDesign Server SDK.

Installing InDesign Server

Before installing InDesign Server, check your system to make sure it matches the relevant system requirements. Then run the installer provided by the ESD (Electronic Software Download). After you install, you can run InDesign Server from the command line, as described in "Running InDesign Server" on page 12. Please note that serialization for InDesign Server occurs at the command line on first launch by using the -serialnumber argument.

System requirements

Please see the detailed system requirements for running InDesign Server, located in the application's ReadMe file.

If you want to run the InDesign Server Windows Service, Microsoft .NET Framework Version 2.0 or higher is required.

If you want to write Java code that uses the InDesign Server Java API, JDK (Java Development Kit) 1.6.0 or higher is required.

Running the installer

Once you download and expand the InDesign Server ESD (Electronic Software Download), you can run the installer by double-clicking the installer icon. Follow the instructions displayed on your screen. During installation, you can specify where to install InDesign Server or use the default location.

On Windows only, when installing to a 64-bit operating system, the 64-bit version of InDesign is installed by default, and you are given the option of also installing the 32-bit version of InDesign Server. When installing to a 32-bit operating system, only the 32-bit version of InDesign Server can be installed.

CJK feature set

InDesign Server performs composition based on the enabled feature set: Roman (English) or Japanese (CJK). On Windows, registry settings are used to specify the enabled feature set. On

Mac OS, there are two applications, one with the Roman feature set enabled and one with the CJK feature set enabled.

When installing InDesign Server on an English version of Windows, the feature set defaults to Roman. When installing on a Chinese version of Windows (for example), the feature set defaults to CJK. During installation on an English Windows, if you set the installation language to Chinese, the feature set defaults to CJK.

You can change the default feature set by running regedit and modifying the following keys (change version number as appropriate):

32-bit application on 32-bit Windows:

`HKEY_LOCAL_MACHINE\SOFTWARE\Adobe\InDesign Server\<version>`

64-bit application on 64-bit Windows:

`HKEY_LOCAL_MACHINE\SOFTWARE\Adobe\InDesign Server\<version>`

32-bit application on 64-bit Windows:

`HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Adobe\InDesign Server\<version>`

Change the Feature Set Locale Setting key to 400 for Roman or 401 for CJK. The setting takes effect when InDesign Server is started.

Firewalls

Any firewall that exists between a machine (server) running InDesign Server and clients of the server must be configured appropriately, so the clients can establish connections with the server. For details, see the documentation for your firewall or consult with your network system administrator.

Installing the InDesign Server Windows Service (Windows only)

The InDesign Server Windows Service monitors specified instances of InDesign Server and restarts them if they terminate or if the machine is rebooted. It redirects InDesign Server messages to the Windows event log, allowing these messages to be viewed locally or remotely, and it allows instances of InDesign Server to be configured with a user name and password separate from that of the machine's current user.

The InDesign Server Windows Service can be installed either by using the InDesign Server installer or from the command line. To use the installer, simply run the InDesign Server installer, and choose the option to install Service Files for InDesign Server. To install from the command line, set your current directory to the folder where you installed InDesign Server, and run the following command:

```
InDesignServerService /install
```

To uninstall InDesign Server Windows Service using the command line, set your current directory to the InDesign Server folder, and run the following command:

```
InDesignServerService /install /u
```

The second component of the InDesign Server Windows Service is the InDesign Server Service Management Console snap-in. If you installed the service using the InDesign Server installer,

the snap-in also was installed then. If you used the command line to install the service, you also need to install the snap-in, as described here.

To install the snap-in on a 32-bit machine, run the following command:

```
regsvr32 InDesignServerMMC.dll
```

On a 64-bit machine, run the following command:

```
regsvr64 InDesignServerMMC64.dll
```

Uninstalling the InDesignServerMMC snap-in is the same as installing, with an additional command-line argument:

```
regsvr32 InDesignServerMMC.dll /u
```

— or —

```
regsvr64 InDesignServerMMC.dll /u
```

For information on configuring the snap-in, see "Configuring InDesign Server Windows Service" on page 17.)

Installing the InDesign Server SDK

The InDesign Server SDK contains samples, scripts, and documentation for InDesign Server. The InDesign Server SDK can be downloaded from Adobe.com. If desired, you can move the SDK folder after setting up the SDK.

Running InDesign Server

You can launch InDesign Server from the command line or from another program or script. Each method allows command-line arguments, which control how InDesign Server runs and is configured.

InDesign Server arguments

The InDesign Server command-line arguments enable you to configure various aspects of InDesign Server. The command line has the following syntax; the arguments are described in Table 1:

```
InDesignServer [arguments]
```

TABLE 1 InDesign Server command-line arguments

Argument	Description

-configuration *name*	Optional. Specifies the configuration name for this instance of InDesign Server. *name* must be a valid folder name in the current system, as InDesign Server uses it to create the configuration folder that stores private data, cache, and configuration files. By default, the folder name is "configuration_noport" or "configuration_*n*" where "*n*" is the port number (if the -port argument is specified). The configuration folder is created in: On Windows:
	`C:\Documents and Settings\<username>\Application Data\Adobe\InDesign Server\<version>\<locale>\`
	On Windows Vista and Windows Server 2008:
	`C:\Users\<username>\AppData\Roaming\Adobe\InDesign Server\<version>\<locale>\`
	Running under Windows Service, the configuration folder is created in: On Windows:
	`C:\Documents and Settings\All Users\Application Data\Adobe\InDesign Server\<version>\<locale>\`
	On Windows Vista and Windows Server 2008:
	`C:\ProgramData\Adobe\InDesign Server\<version>\<locale>\`
	On Mac OS, the configuration folder is created in:
	`/Users/<username>/Library/Preferences/Adobe InDesign Server/<version>/<locale>/`

-[no]console	Optional. Windows only. This creates a new console window (a command shell) and redirects stderr and stdout there. If -noconsole specified, stdout and stderr are piped to the terminal (or window) that launched InDesign Server. This argument is useful when using the Microsoft Visual Studio debugger to debug an InDesign Server plug-in that you are developing, because by default, the debugger discards data written to stderr and stdout. This also might be useful when starting InDesign Server from within another application, like an application server. This argument can be used only with InDesignServer.exe, not with InDesignServer.com. The default is -noconsole.
-[no]errorlist	Optional. Controls whether InDesign Server should store the scripting errors from SOAP invocations in memory. The error list is not cleared automatically; if more errors are accumulated than allowed for, new errors are lost. The default is -noerrorlist.
-help	Optional. Displays short descriptions of all arguments.
-iorfile *filepath*	Required for use with Java/CORBA; otherwise, optional. This argument tells InDesign Server where to write the file containing the application object's IOR. The IOR in this file is used by a client to create a reference to the InDesign Server application object. You can think of this as a cookie that your client code uses to access InDesign Server's application object. Each server instance requires a unique IOR file. Multiple clients access a given server using the IOR file for that server. You will know that the IOR was written to the specified file if InDesign Server outputs the following notification: "[server] Writing IOR to ..."

Argument	Description
-LogToApplicationEventLog	Optional. On Windows, this argument routes InDesign Server's console output to the Window's application event log. The output can be viewed in the standard Application Event Viewer Computer Management Control Panel. On Macintosh, this argument should be used when InDesign Server is used as a launchd daemon. To view the output, use the standard Mac OS Console application.
-maxerrors *number*	Optional. *number* specifies the maximum number of scripting errors from SOAP invocations that InDesign Server should store in memory. The default is 10,000.
-pluginpath *path*[,*path*]	Required for use with Java/CORBA; otherwise, optional. This argument directs the server to load all plug-ins in the specified folder(s) and their subfolders. By specifying the installed server/corba subdirectory, the InDesign Server Corba Support plug-in is loaded and enabled. *path* is a relative path based on the InDesign Server install folder (for example, passing in "Server/Corba" specifies the folder "C:\Program Files\Adobe\ Adobe InDesign CS5 Server\Server\Corba"). Do not use an absolute path. You will know that the Corba Support plug-in loaded if InDesign Server outputs the following notification: "[server] ApplicationIOR: ..."
-port *number*	Required for use with SOAP; otherwise, optional. This is the port number of the socket that InDesign Server will use to service SOAP requests from client applications. If not specified, no port is used, and SOAP requests cannot be processed. Each instance of InDesign Server running on the same system must have a unique port when enabling SOAP. The port number must be a positive integer value.
-[no]previews	Optional. Specifies whether to generate previews when importing images. The default is -nopreviews.
-[no]seh	Optional. Windows only. Specifies whether structured exception handling should be used. The default is -seh.
-[no]sendcrashlogs	Mac OS only. Automatically sends crash logs to Adobe. The default is -nosendcrashlogs.
-serialnumber *number*	Enters the serial number for the application. This argument needs to be used only once to unlock the application.

Starting InDesign Server from the command line

1. On the system where InDesign Server is installed, open a new shell window. On Windows, use the command prompt; on Mac OS, use Terminal.
2. Change the current directory to the folder where InDesign Server is installed. On Windows, the folder contains the InDesignServer.com application. On Mac OS, the folder contains the InDesignServer.app application package. For example:

```
Windows:
cd "c:\Program Files\Adobe\Adobe InDesign CS5 Server\"
Mac OS:
cd '/Applications/Adobe InDesign CS5 Server/'
```

3. Run the following command. (On Windows, InDesignServer.com is run by default. It sets up paths and executes InDesignServer.exe.)

```
InDesignServer
```

4. InDesign Server displays its start-up messages in the shell window. When InDesign Server is ready for use, it displays "[server] Server Running." While InDesign Server is running, messages and errors are displayed in the shell window, so you can use this window to monitor InDesign Server.

Starting InDesign Server for use with SOAP

If your workflow involves using scripts to interact with InDesign Server, that workflow most likely involves sending the scripts to InDesign Server via SOAP. To communicate with InDesign Server via SOAP, you must use the -port argument when starting up InDesign Server. In the following sample command line, port number '12345' can be replaced with your desired port.

```
InDesignServer -port 12345
```

As InDesign Server displays its start-up messages, you should see this message:

```
"[soap] Servicing SOAP requests on port 12345."
```

This indicates InDesign Server is ready to accept SOAP requests.

Starting InDesign Server for use with CORBA

If your workflow involves using Java to interact with InDesign Server, you must start InDesign Server for use with CORBA. The InDesign Server Java API relies on the Corba Support plug-in to communicate with InDesign Server. This communication is facilitated through the use of a text file that contains a string representing an IOR that uniquely identifies an object in CORBA.

Once the InDesign Server instance is initialized, it writes the IOR representing its application object to the specified text file. Using the IOR contained in the IOR file, a client can instantiate a proxy object for the InDesign Server application object.

To start InDesign Server for use with CORBA, add the -iorfile and -pluginpath arguments to your command line. The path specified by -pluginpath is relative to the path of the InDesign Server application. At start-up, InDesign Server creates the specified text file and writes its application's IOR to the file. You must have write permission to the file path. In the following sample command line, replace c:\ior.txt (Windows) or ~/ior.txt (Mac OS) with your desired filepath.

Windows:

```
InDesignServer -iorfile c:\ior.txt -pluginpath Server\Corba
```

Mac OS:

```
InDesignServer -iorfile ~/ior.txt -pluginpath Server/Corba
```

As InDesign Server displays its start-up messages, you should see the following messages:

```
"[server] ApplicationIOR:..."
```

```
"[server] Writing IOR to ..."
```

These indicate that InDesign Server is ready to accept CORBA commands and you can access the InDesign Server application IOR from the specified file.

Starting from another program or script

InDesign Server can be launched as a process from another program or script. On Windows, be aware that InDesign Server typically is run using InDesignServer.com rather than InDesignServer.exe. The .com version differs from the .exe in that it sets up required paths to the omniorb folder required by the InDesign Server CORBA implementation. The .exe does not set up these paths, so you may need to add the omniorb folder to your PATH environment variable before launching InDesignServer.exe.

Running multiple InDesign Server instances

If your site has a high volume of InDesign Server requests, you can increase throughput by running multiple instances of InDesign Server on the same system. A general rule of thumb for the maximum number of InDesign Server instances running on one machine is to add one to the number of cores on the machine. For more information, see *Adobe InDesign Server CS5 Performance and Scalability* in the InDesign Server SDK.

To run multiple instances, you must give each instance a unique configuration name. Do this with the -configuration argument, specifying a unique name for each instance. When running InDesign Server from the command line, you must run each instance in its own window.

From a shell window, start the first instance by executing the following line:

```
InDesignServer -configuration myIDS1
```

Then, open a new shell window and execute the following line for the second instance:

```
InDesignServer -configuration myIDS2
```

Multiple instances using SOAP

If you have applications that communicate with InDesign Server using SOAP, you must assign a different port number to each instance of InDesign Server. In the following examples, you do not need to specify the -configuration argument, because InDesign Server creates a configuration name based on the specified port (as described in "InDesign Server arguments" on page 12). The following examples create the configuration names "configuration_18383" and "configuration_18555."

The first instance is run by executing the following command line:

```
InDesignServer -port 18383
```

A second instance is run by executing the following command line:

```
InDesignServer -port 18555
```

Optionally, you can give each instance a unique configuration name; however a unique port number is still required for using SOAP. For example:

```
InDesignServer -configuration myIDS3 -port 18777
```

Multiple instances using CORBA

If you have applications that communicate with InDesign Server using Java/CORBA, you must assign each instance of InDesign Server a different IOR text file and configuration name. The following examples create unique IOR text files and configuration names for each instance.

On Windows, the first instance is run by executing the following command line:

```
InDesignServer -configuration myIDS1 -iorfile c:\IDS\ior_1.txt
     -pluginpath Server/Corba
```

A second instance is run by executing the following command line:

```
InDesignServer -configuration myIDS2 -iorfile c:\IDS\ior_2.txt
     -pluginpath Server/Corba
```

On Mac OS, the first instance is run by executing the following command line:

```
InDesignServer -configuration myIDS1 -iorfile /IDS/ior_1.txt
     -pluginpath Server/Corba
```

A second instance is run by executing the following command line:

```
InDesignServer -configuration myIDS2 -iorfile /IDS/ior_2.txt
     -pluginpath Server/Corba
```

Quitting InDesign Server

You can quit InDesign Server in several ways:

- Press Control+C in the Command Prompt window (Terminal on Mac OS) where InDesign Server was started.
- Send a script to InDesign Server telling it to shut down using the application's quit event. For example, the following JavaScript quits InDesign Server without saving open documents:

  ```
  app.quit(SaveOptions.no);
  ```

- If the InDesign Server instance was launched from the InDesign Server Windows Service, you can stop the service to terminate all instances of InDesign server started by the service.

Configuring InDesign Server Windows Service

The InDesign Server Windows Service can be started, stopped, and configured through the "Services" category of the standard Microsoft Management Console. This can be launched either by right-clicking My Computer > Manage or through the Start menu, Start > Administrative Tools > Computer Management. In the Computer Management Window, choose "Services and Applications", then "Services." In the list of services, double-click InDesignServerService. This brings up a properties dialog that allows you to start, stop, or pause the service, modify the login, or change how the service starts.

On the Log On tab, Microsoft recommends using the least privileged setting that allows the service to do what it needs to do. In this case, that means reading and writing to the "Documents and Settings" folder and to the location of the documents the server is acting on; opening a port; and reading registry settings. "Local System account" is more privileged than the

LocalService, so you should first attempt to set "this account" to a LocalService (for example, "NT AUTHORITY\LocalService").

On the Recovery tab, we recommend that you leave the settings as "Take no action." This provides the option of restarting the service if it crashes. If the service is doing its job, it has already launched instances of InDesignServer with specified ports/configurations. If the service then crashes, and the InDesign Server instances are still running, the new instance of the service will not know about those instances and will try to start new instances of InDesign Server; those will fail to launch, because the ports/configurations already are in use. These settings do not affect how the service handles instances of InDesign Server if they crash; that is configured via the registry settings MaximumFailureCount, MaximumFailureIntervalInMinutes, and TrackFailures in HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\InDesignCS5ServerWinService.

InDesign Server Console Management snap-in

The second component of the Windows Service is the InDesign Server Console Management snap-in. To add the snap-in to a management console, double-click the InDesignServerService.msc file found in your InDesign Server installation. You also can manually add the snap-in to a management console, by running the Windows application mmc.exe and choosing File > Add/Remove Snap-in... This brings up a list of all available MMC snap-ins. If no list is visible, click Add. Choose InDesignCS5ServerService from the list, to add it to the current console. Close the Add/Remove Snap-In dialog.

To add an InDesign Server instance to the console, right-click InDesignCS5ServerService in the left pane of the console, and choose New > New InDesign Server instance. The new instance inherits the failure parameters from the last instance. It is assigned a port number one greater than the highest port number among existing InDesignCS5ServerService instances. The default command line is empty. To modify an instance's configuration (port number or command-line), double-click the instance in the right pane. The command line should be formatted as described in "Starting InDesign Server from the command line" on page 14, with the exception that no -port argument should be used, as it is defined by the port entry.

For multiple instances of InDesign Server to exist on one machine, each instance must have a unique port or unique configuration name; these can be specified by editing the port and command-line settings for the service. Although usually you need to edit only the port and command-line settings for the service, there are a few other settings that can be modified by editing the Registry Settings for the service instance; these settings include:

- MaximumFailureCount — The number of times the service tries to restart a failed instance.
- MaximumFailureIntervalInMinutes — The timeframe for MaximumFailureCount. InDesign Server must fail more than MaximumFailureCount times within MaximumFailureIntervalInMinutes, before the service stops trying to restart it.
- TrackFailures — This is a Boolean. If it is 1 (the default), the service tries to restart InDesign Server; if 0, the service does not.

To edit the settings, run regedit and look at the keys in HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\InDesignCS5ServerWinService. There should be one set of keys (for example, 065c361a-0cbe-4917-bf15-5276b1815769) for each instance you configured using InDesignServerService.msc or the Console Management snap-in.

Configuring an InDesign Server launchd daemon for Mac OS

InDesign Server can be launched as a background task (daemon) on Mac OS, without requiring direct control by an end user. Running InDesign Server as a daemon allows you to configure InDesign Server to launch when the operating system starts and offers the ability to configure restart options if the application crashes.

To configure the daemon, you create a properties list (plist) configuration file containing the desired settings and install it in the desired LaunchAgents or LaunchDaemons folder.

The launchd Configuration file

To create a launchd configuration file, use your favorite plain text editor, and enter the XML content as specified below. Save the file using your daemon's Label as the filename, and .plist as the extension (for example, com.adobe.ids.launchd.12345.plist). You also can edit a plist file using the XCode utility application, Property List Editor, located in Developer/Applications/Utilities.

Table 2 lists the required settings for an InDesign Server launchd daemon. For a complete list of launchd options, see the man page for launchd.

TABLE 2 InDesign Server daemon settings

Label	The unique identifier for your daemon; for example, com.adobe.ids.launchd.12345
OnDemand	This must be set to *false* for InDesign Server to restart following a quit or crash.
ProgramArguments	This is an array of strings representing the arguments used when launching the application. The first element of the array is the full path to the application. The remaining array elements are the same arguments used when starting InDesign Server from the command line. To receive console output from InDesign Server, you must use the -LogToApplicationEventLog argument.
RunAtLoad	This is set to *true*. InDesign Server is started when the daemon is loaded at System Startup or loaded manually using launchctl.

EXAMPLE 1 xml format of plist

The following is the XML content of a plist configuration file that will run InDesign Server using SOAP port 12345:>

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd"
<plist version="1.0">
<dict>
      <key>Label</key>
      <string>com.adobe.ids.launchd.12345</string>
```

```
        <key>OnDemand</key>
        <false/>
        <key>ProgramArguments</key>
        <array>
            <string>/Applications/Adobe InDesign CS5 Server/InDesignServer</string>
            <string>-port</string>
            <string>12345</string>
            <string>-LogToApplicationEventLog</string>
        </array>
        <key>RunAtLoad</key>
        <true/>
</dict>
</plist>
```

Install the configuration file

To install a launchd configuration file, you simply copy it to the correct folder on the machine where InDesign Server will be run. You can have multiple daemon configuration files, one for each instance of InDesign Server you want to run. For a typical InDesign Server installation, you will run Mac OS X Server with no user accounts. In this case, you copy the configuration file to:

```
/Library/LauchDaemons
```

If want to run InDesign Server from a user account, copy the plist file to the LaunchAgents folder in either the system Library folder or the user's Library folder (you may need to create the user's LaunchAgents folder):

```
/Library/LaunchAgents
/Users/<username>/Library/LaunchAgents
```

Load the daemon

After the plist file is copied to the launch folder, restart the system to automatically load the daemon. After the daemon is loaded, InDesign Server is launched using the settings specified in the plist file. To confirm that the daemon was loaded, use the following command from a Terminal window:

```
launchctl list
```

If you see your daemon's Label listed, you know it was loaded correctly.

To confirm that InDesign Server is running, you can use Activity Monitor to look at all running processes.

You can manually load the daemon using the launchctl command:

```
launchctl load /Library/LaunchDaemons/com.adobe.ids.launchd.12345.plist
```

Unload the daemon

You can manually unload the daemon using launchctl, but be aware that unloading the daemon will not force the InDesign Server instance to quit:

```
launchctl unload /Library/LaunchDaemons/com.adobe.ids.launchd.12345.plist
```

To quit the InDesign Server instance, send a script to InDesign Server telling it to quit, or use Activity Monitor to quit the process.

Interfacing with InDesign Server through SOAP

InDesign Server supports one SOAP command, *RunScript*, which allows you to run an InDesign Server script. You can write your script using JavaScript, VBScript (Windows only), or AppleScript (Mac OS only). When you call RunScript, it passes your script file to InDesign Server, and InDesign Server executes the script internally.

The RunScript command is defined by the InDesign Server WSDL. You can use this WSDL with industry-standard SOAP tools to generate SOAP message packets automatically. There are two ways to view the WSDL:

- Look at the InDesign Server SDK file, IDSP.wsdl.
- Access the WSDL through HTTP after starting InDesign Server using the -port argument (for example, http://localhost:12345/service?wsdl).

For information on how to write a SOAP client that interacts with InDesign Server, see *Working with Adobe InDesign CS5 Server SOAP* or any of the "sampleclient" samples in the SDK.

Using sampleclient to run an InDesign Server script

InDesign Server provides a sample SOAP client application, *sampleclient*. sampleclient is a command-line application that sends a script to InDesign Server via the InDesign Server *RunScript* SOAP command. The script can be written in JavaScript, AppleScript, or VBScript. SampleClient assumes that the extension for an AppleScript is .applescript; for a VBScript, .vbs. Any other extension is assumed to be JavaScript.

The InDesign products SDK includes the components necessary to construct sampleclient. The project file is in <*SDK*>/build/<*platform*>/prj, and the source code is in <*SDK*>/source/public/components/server/sampleclient.

sampleclient command-line arguments

The sampleclient arguments identify the system on which InDesign Server is running, the port configured to accept SOAP communication, the name of the script file you want to run, and any arguments to be passed in to the script. sampleclient has the following syntax; the arguments are described in the following table:

```
sampleclient [-host hostID:port] [-server] scriptPath [arguments]
     [-repeat number_of_reps] [-h]
```

TABLE 3 sampleclient command-line arguments

Argument	Description

arguments	Optional. This is one or more name-value pairs used to pass arguments to the script. The format is *argname=value*, where *argname* is the name of the argument the script reads, and *value* is its value. Separate multiple arguments with spaces. For example:
	`sampleclient -host localhost:18383 /myScript.jsx myArg="hello world"`
	Scripts can read the argument value by using the get method of the scriptArgs object in the application object. For example, the following gets the argument named *myArg*:
	`myArg = app.scriptArgs.get("myArg");`

-h	Optional. Displays short descriptions of these arguments.

-host *hostID:port*	Optional. The host on which InDesign Server is running. *hostID* can be one of the following identifiers for the system on which InDesign Server is running: the IP address of the system, the host name of the system (for example, ourhost.corp.company.com), or the keyword "localhost" if InDesign Server is running on the same machine as sampleclient. Both IPv4 and IPv6 addresses are accepted. *port* is the host port number that is configured to accept SOAP messages. For example:
	`-host 10.0.0.1:18383`
	`-host [fe80::405:2345:fe56:8901]:18383`
	On a local system:
	`-host localhost:18300`

-repeat	Optional. Specifies the number of times to execute the script.

Argument	Description

scriptPath	Required. Without the -server argument, *scriptPath* is the path to the script file on the local file system. The path can be an absolute path or a relative path based on the current working directory. If the -server argument is used, *scriptPath* can take one of two forms. First, *scriptPath* can be an absolute path to the script based on the file system of the targeted InDesign Server instance. On Windows:
	`C:\<myScriptsFolder>\myScript.jsx`
	On Mac OS:
	`~/<myScriptsFolder>/myScript.jsx`
	
	Second, *scriptpath* can be based on a predefined path constant. InDesign Server defines two path constants, Scripts:Application and Script:User. Scripts:Application represents the application's scripts folder. It is located: On Windows:
	`C:\Program Files\Adobe\Adobe InDesign CS5 Server\Scripts\`
	On Mac OS:
	`/Applications/Adobe InDesign CS5 Server/Scripts/`
	
	Scripts:User represents the user's scripts folder. It is located: On Windows:
	`C:\Documents and Settings\<username>\Application Data\Adobe\InDesign Server\<version>\<locale>\configuration_12345\Scripts\`
	On Windows Vista and Windows Server 2008:
	`C:\Users\<username>\AppData\Roaming\Adobe\<version>\<locale>\configuration_12345\Scripts\`
	On Mac OS:
	`/Users/<username>/Library/Preferences/Adobe InDesign Server/<version>/<locale>/configuration_12345/Scripts`
	
	Place your script file in one of the scripts folders on the InDesign Server machine, then pass the corresponding path to sampleclient. For example (note that you must use a colon (:) as the path separator on both Windows and Mac OS):
	`Scripts:Application:myScript.jsx`
	`Scripts:User:myScript.jsx`

-server	Optional. If this keyword is specified, *scriptPath* is assumed to represent a file on the machine where InDesign Server is running. Otherwise, *scriptPath* is assumed to represent a file on the local machine, where SampleClient is running.

Starting SampleClient

1. You need a script file that can be executed by InDesign Server. You can use one of the sample scripts from the InDesign Server SDK or write your own script using the InDesign scripting API. If you write your script, keep in mind that InDesign Server does not have a user interface, so make sure your script does not rely on any user-interface components.
2. Launch InDesign Server for use with SOAP. For details, see "Starting InDesign Server for use with SOAP" on page 15. Be sure to note the host and port on which InDesign Server is launched.
3. Open a new command-line window (on Windows, use command prompt; on Mac OS, use Terminal).

4. Change directories to where SampleClient is located:

```
Windows:
   cd "c:\Program Files\Adobe\Adobe InDesign CS5 Server"

Mac OS:
   cd '/Applications/Adobe/Adobe InDesign CS5 Server'
```

5. Run the SampleClient command with the appropriate host, port, and script file path. See the following examples:

```
If InDesign Server is running locally:
   sampleclient -host localhost:18383 yourScriptPath

If InDesign Server is running on a different system:
   sampleclient -host 192.168.1.100:18383 yourScriptPath
```

6. sampleclient displays messages and the return value for the script in the command-line window, so you can use it for monitoring.

Interfacing with InDesign Server through Java

InDesign Server provides a Java API that exposes the InDesign Server scripting DOM (Document Object Model). Communication between the Java API and the scripting DOM is facilitated through CORBA. When you execute your Java code, the Java API makes calls to the CORBA API. The CORBA API in turn sends commands to the InDesign Server C++ code that exposes the scripting DOM.

The InDesign Server Java API is distributed in the InDesign Server SDK, in the form of a JAR file. This JAR file is required to compile and execute InDesign Server Java code. While the API is based on CORBA, it explicitly targets a Java audience.

For information on how to write a Java component that interacts with InDesign Server, see *Working with Adobe InDesign CS5 Server Java*.

Write a Java component

The first step in accessing InDesign Server through CORBA is to write a Java component using the InDesign Server Java API. The API provides full access to the InDesign Server scripting DOM; each component of the InDesign Server scripting DOM is represented as a Java class. These classes are then packaged together into InDesignServerAPI.jar.

The InDesign Server SDK contains the InDesign Server API JAR file (<*IDS SDK*>/lib/InDesignServerAPI.jar) and many Java code samples (<*IDS SDK*>/samples/snippets) demonstrating the use of the InDesign Server Java API. *Working with Adobe InDesign CS5 Server Java* explains how to compile and run the samples.

Running an InDesign Server Java application

There are several ways to launch a Java application to work with InDesign Server, including the "java" tool on the command line, a launcher Java application, and a Web page with a Java applet. The method discussed here uses the command line:

1. Launch InDesign Server, and make sure it is set up to accept CORBA commands. For details, see "Starting InDesign Server for use with CORBA" on page 15. Be sure to note the IOR file path your InDesign Server instance is using.
2. Open a new command-line window (on Windows, use the command prompt; on Mac OS, use Terminal).
3. Use the java tool on the command line to run your code. The -classpath argument adds the specified .jar file(s) and/or directories to the classpath. The classpath is where the JRE searches for the class being invoked. The second argument is the name of the class whose main() method you want to run. If this class is contained within a package, prepend the package to the class name. In the example below, the package is com.adobe.ids.sdk. The third argument, ior.txt, is an argument to the main() method being called. In this case, it is the filepath to the IOR file InDesignServer is using.

```
java -classpath "..\lib\InDesignServerAPI.jar;..\lib\samples.jar"
com.adobe.ids.sdk.mySampleClass "ior.txt"
```

Interfacing with InDesign Server through a plug-in

Developing a plug-in for InDesign Server requires the same fundamental C++ and object-oriented programming and design skills needed to develop plug-ins for InDesign. Because InDesign Server does not have a user interface like its desktop counterpart, there are many special considerations when creating a new plug-in or porting an existing InDesign plug-in for use under InDesign Server.

Adobe InDesign CS5 Server Plug-in Techniques discusses how to write plug-ins that run safely in the InDesign Server environment. This document also discusses how to build 64-bit plug-ins.

For your plug-in to load and run under InDesign Server x64, you must convert your plug-in to a 64-bit version. You can run your 32-bit plug-in on a 64-bit machine by running the 32-bit version of InDesign CS5 Server. 32-bit applications can run on 64-bit platforms in the emulation mode called WOW64 (Win32 On Win64); however, 64-bit applications cannot run on 32-bit platforms.

To interact with your plug-in through scripting or Java, you need to make your plug-in scriptable. For details, refer to the "Scriptable Plug-in Fundamentals" chapter in the *Adobe InDesign CS5 Programming Guide*; it explains how to expose your plug-in's API in the scripting DOM.

If you will access your plug-in from a Java component, you also need to regenerate both InDesignServerAPI.jar and the InDesign Server Corba Support plug-in file. Regenerating these files will add your plug-in's functionality to those components. The process for regenerating the Java

API JAR file and CORBA Support plug-in is explained in *Regenerating the Adobe InDesign CS5 Server Java API*.

Note that the InDesign Server SDK provides a single version of InDesignServerAPI.jar that works with both 64-bit and 32-bit InDesign Server.

Interfacing with InDesign Server through COM (Windows only)

InDesign Server publishes a COM type library that can be used to create COM components that interoperate with InDesign Server. The type library contains definitions of the classes, methods, and constants of the InDesign Server scripting DOM.

InDesign Server is plug-in based, so its scripting DOM changes depending upon which plug-ins are available at start-up. To remain valid, the InDesign Server type library is created or updated on start-up and stored on the local hard drive. The resulting .tlb file is saved to the Application Data folder:

```
C:\Documents and Settings\All Users\Application Data\Adobe\InDesign
Server\<version>\<locale>\configuration_12345\Scripting Support\<version>\Resources
for Visual Basic.tlb
```

(or on Windows Vista and Windows Server 2008):

```
C:\ProgramData\Adobe\InDesign
Server\<version>\<locale>\configuration_12345\Scripting Support\<version>\Resources
for Visual Basic.tlb
```

In the preceding path, "configuration_12345" is the configuration name for the InDesign Server instance that created the table file.

Because the InDesign Server COM type library is based on the loosely typed InDesign Server scripting model, it also is loosely typed. When you choose a language to develop your COM component, keep in mind that the more loosely typed the language, the more compatible it is with the InDesign Server COM type library.

Visual Basic

Your best bet for creating an InDesign Server COM component is to 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.

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. You also should 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.

To use the InDesign Server COM type library in your project, you need to have access to the interop DLL for InDesign Server. Under .NET, you can do this by making a reference to the .tlb file. Right-click on your project and choose "Add Reference." Then choose the COM tab and locate "Adobe InDesign CS5 Server Type Library." If it does not appear in the dialog, you first need to launch InDesign Server on your machine, so InDesign Server will create the type library.

To create a reference to an instance of InDesign Server in your code, use the CreateObject method. Note that CreateObject does not give you control over which InDesign Server instance the script will target.

```
Set myApp = CreateObject("InDesignServer.Application")
```

To create a reference to a specific instance of InDesign Server in your code, use the GetObject method, passing the configuration name of the InDesign Server instance you want to target:

```
Set myApp = GetObject("configuration_12345")
```

For more information and samples, examine the <*IDS SDK*>/samples/ folder, and read *Adobe InDesign CS5 Scripting Guide*.

C#

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. That being said, it is possible to write simple InDesign Server COM components with careful type-casting and parameter specification.

One possible solution is to have your C# code call the doScript method of InDesign Server, passing in a script file or the text of a script. For an example, see the sampleclient-csharp-com sample in the InDesign Server SDK.

When trying to create a more involved component with C#, you may encounter pitfalls including the following:

- C# is strongly typed, so it always must know a type for an object. This means you must typecast nearly every object retrieved from the scripting DOM. This is fine if you are sure what type your object is; however, many methods in the scripting DOM accept or return variable types, and you may not be able to predict the returned type.
- You must specify all parameters to methods, even though they may be optional in the scripting DOM. You cannot pass "null"; parameters must be fully formed objects.
- Optional parameters may be impossible to deal with. One example is where a method in the scripting DOM allows you to *not* pass the first parameter of a method, in which case InDesign Server pays attention to the second parameter. This is impossible under C#.

To create a reference to an instance of InDesign Server in C#, use the InteropServices method Marshal.BindToMoniker:

```
myApp = (InDesignServer.Application)
(System.Runtime.InteropServices.Marshal.BindToMoniker("configuration_12345"));
```

For details and samples, examine the InDesign Server SDK Samples folder.

Interfacing with InDesign Server through LBQ

InDesign Server includes an optional component called LBQ that provides job queueing and load balancing functionality for InDesign Server. LBQ is strictly optional. It is not required for running InDesign Server. It is one solution for load balancing and job queueing.

LBQ provides load balancing across multiple instances of InDesign Server running on one or more machines. It uses JAVA and CORBA to communicate with the instances of InDesign Server, so each instance must be started as described in "Starting InDesign Server for use with CORBA".

In addition to load balancing and basic queueing, LBQ supports the following features:

- Prioritized queueing
- Status visibility
- Logging
- Multiple queues
- Editing queued jobs

Clients interact with LBQ using a RESTful API. RESTful API calls are made through a URL request. Upon executing a request, an XML response is returned.

LBQ supports the following operations:

- EnqueueJob
- ModifyJob
- CancelJob
- StartQueue
- StopQueue
- ReinitializeQueue
- GetVersion
- JobStatus
- QueueStatus
- IDSStatus

Refer to the document *Working With Load Balancing and Queueing For Adobe InDesign CS5 Server* for detailed explanations of these operations as well as other helpful information about working with LBQ.

Handling error messages

InDesign Server logs its messages and errors to stderr and stdout. These errors and messages can be captured or redirected.

Accessing errors and messages

From a plug-in, use the C++ APIs MessageLog and/or IErrorList. This is explained in *Adobe InDesign CS5 Server Plug-in Techniques*.

From the Java API, use the ErrorListError objects contained within the application object.

Redirecting errors and messages

When running InDesign Server from within an application, you can configure stdout and stderr to convey messages from InDesign Server to your application.

In Windows, InDesign Server messages can be redirected to the Window's application event log by launching InDesign Server with the -LogToApplicationEventLog argument. If you are running InDesign Server from the InDesign Server Windows Service, logged messages are redirected automatically to the Window's application event log. The event log can be viewed in the standard Event Viewer, in the Computer Management Control Panel.

On Mac OS, you can redirect messages to the system console by launching InDesign Server with the -LogToApplicationEventLog argument. The log can be viewed using the utility application, Console.

Next steps

After learning to run InDesign Server and using SOAP or Java to run a sample provided in the SDK, 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.

InDesign Server scripting, plug-ins, and Java

For information on writing InDesign and InDesign Server scripts, plug-ins, and Java components, install the InDesign products SDK and InDesign Server SDK, then reference the following:

- *Adobe InDesign CS5 Server Performance and Scalability* — Provides benchmarks and information on how to configure your system to maximize the performance and scalability of InDesign Server.
- *Adobe InDesign CS5 Programming Guide* — Provides information about creating C++ plug-ins for InDesign Server.

- *Adobe InDesign CS5 Scripting Guide* — Provides basic information about, and numerous examples of, creating scripts for InDesign. There are three versions of this document, for AppleScript, JavaScript, and VBScript.
- *Adobe InDesign CS5 Server Java API Reference* — A javadoc reference of the InDesign Server Java API.
- *Adobe InDesign CS5 Server Plug-in Techniques*.
- *Regenerating the Adobe InDesign CS5 Server Java API*.
- *Working with Adobe CS5 InDesign Server Java*.
- *Working with Adobe CS5 InDesign Server SOAP*.
- *Adobe Scripting*, by McWilliams Chandler. New York: Wiley Publishing Inc., 2003. ISBN 0-7645-2455-0.

JavaScript

For documentation on the JavaScript language and descriptions of how to use it, see the following:

- *JavaScript: The Definitive Guide, 4th Edition*, by Flanagan, David. O'Reilly, 2001. ISBN 0-596-00048-0.
- *JavaScript Bible. 5th Edition*, by Danny Goodman and Michael Morrison. John Wiley and Sons, 1998. ISBN 0-7645-57432.
- *JavaScript Programmer's Reference*, by Cliff Wooton. Wrox, 2001. ISBN 1-861004-59-1.

SOAP

For information on SOAP communication, see the SOAP primer from W3C, located at http://www.w3.org/TR/soap12-part0/.

Java

For documentation on the Java language and descriptions of how to use it, see the following:

- *Java in a Nutshell, 5th edition*, by David Flanagan. O'Reilly Media, 2005. ISBN 0-596-00773-6
- http://java.sun.com

CORBA

For documentation on CORBA and descriptions of how to use it, see the following:

- http://www.omg.org/gettingstarted/corbafaq.htm
- http://omniorb.sourceforge.net/ — Provides information about the ORB implementation InDesign Server uses, OmniORB.

Frequently asked questions

Can I specify both a SOAP port and an IOR file when launching the server?

Yes. InDesign Server can handle SOAP and CORBA commands concurrently; however, you may experience performance degradation.

Is there an index of all objects and methods in the InDesign Server Java API?

Yes. A Java API reference is included in the InDesign Server SDK.

Is there an index of all classes and methods in the InDesign Server scripting DOM?

Yes. An Object Model Reference for InDesign Server can be accessed from the Help menu inside Adobe ExtendScript Toolkit CS5.

How do I make calls to my own plug-in from the InDesign Server Java API?

You need to regenerate the InDesignServerAPI.jar file and Corba Support plug-in. For information on how to do this, see *Regenerating the Adobe InDesign CS5 Server Java API*, included in the InDesign Server SDK

Can I interact with InDesign Server if I launch it for neither SOAP nor CORBA?

Yes. You can communicate with InDesign Server through AppleScript, VBScript, or COM, without using SOAP or CORBA.

How do I return a value from my script to my SOAP client?

In JavaScript and AppleScript, the script's return value is the last value encountered in the script. In VBScript, to define the return value for the script, you set a variable named returnValue. Each of the following examples returns the name of the document at the first index.

JavaScript:

```
var documentName = app.documents.item(0).name;
documentName;
```

AppleScript:

```
tell application "InDesignServer"
     set documentName to name of document 1
end tell
```

```
documentName
```

VBScript:

```
Set myApp = CreateObject("InDesignServer.Application.CS5")
documentName = myApp.Documents.Item(1).Name
returnValue = documentName
```