

Adobe Document Server

QuickStart Guide

QuickStart Guide

Contents

Part 1: Getting Started	2
Part 2: Installation	4
Part 3: Creating documents with XML commands and Adobe Document Server	8
Part 4: Communicating to Adobe Document Server with an API	18



Adobe Document Server:

QuickStart Guide

This QuickStart guide provides the instructions needed to get your Adobe Document Server (ADS) running quickly—complete with installation requirements and instructions, sample code, and general descriptions of this product. This guide covers the PDF manipulation capabilities of Adobe Document Server, and is targeted to developers with XML experience, as well as expertise in Java, COM, or Perl. If you are interested in the ability to create new images or dynamic graphs, see the Adobe Graphics Server QuickStart Guide. (Information on creating documents using XSL-FO and FrameMaker templates is outside the scope of the document.)

Adobe Document Server modifies and creates documents as part of an automated document creation workflow. Requests to Adobe Document Server can be executed via XML command requests using any of the following five interfaces: command line interface, Java API, Perl API, COM API, or directly to the ADS Web Service.

(For more information, see the complete documentation for the Adobe Document Server, located in the ADS installation directory. You can also find valuable information on the Adobe Web site at www.adobe.com. The Web site includes links to user-to-user forums and the Developer Knowledge Base.)

Part 1: Getting Started

Adobe Document Server runs on Solaris and Windows platforms. Prior to installation of Adobe Document Server, check that your system meets the following system requirements.

Windows system requirements

- Microsoft Windows 2000 Server with Service Pack 2 or Microsoft Windows NT® 4.0 Server with Service Pack 6a and Windows Installer update
- Intel® Pentium® III 500 MHz or faster processor
- 512 MB RAM per CPU (1 GB per CPU recommended)
- Swap disk space: RAM size plus 256 MB
- 350 MB available disk space for installation
- CD-ROM drive

Before installation, determine which API you will use with Adobe Document Server. Adobe Document Server supports Java, Perl, COM, or the Web service. If you want to use the COM API and have questions regarding the configuration of the COM API or other COM+ features, see the “Configuring the COM+ Application” in the “Installation and Configuration Guide” found in the documentation folder of your installation CD.

If you want to use the Web service and are interested in communicating to the Web Service directly via SOAP, see the “Adobe Document Server API Reference.” (A description of the SOAP API is outside the scope of this document because the Java and Perl SDKs abstract the specifics of the SOAP API away.)

Solaris system requirements

- Sun® Solaris 7 or 8
- Sun UltraSPARC® Ili 440 MHz or faster such as the Ultra 10
- 512 MB of RAM per CPU (1GB per CPU recommended)
- Swap space: RAM size plus 256 MB
- 350 MB of available disk space for installation
- CD-ROM drive

Before you install the Adobe Document Server for Solaris, determine which Solaris package you need for your application requirements. To use the command line version of Adobe Document Server, install the ADBdocsvr Package. To use the Java or Perl API, install the corresponding SDK: ADBjavadk Package or ADBperlDK Package. To ensure that all samples function properly with this installation, install both ADBdocsvr and ADBjavadk packages.

Java Development Kit (JDK)

To use the Adobe Document Server Java SDK, install the Java Development Kit (JDK) Java 2 version 1.2.2 or later. The JDK is available to download from the Sun® Web site at <http://java.sun.com>. The JDK includes the Java Runtime Environment (JRE), which you need to view the Java samples in the Adobe Document Server Java SDK. The JRE is also available from the Sun Web site at <http://java.sun.com>.

Perl

To use the Adobe Document Server Perl SDK, install a Perl 5 interpreter to run the Perl samples and use the Perl APIs. Adobe recommends the Perl interpreter ActivePerl® 5.6, which is available from the ActiveState® Web site at <http://www.activestate.com>.

Part 2: Installation

You can install the Adobe Document Server software (including samples and documentation) on Windows or Solaris. After you install the software, you can verify that all components installed correctly, and you can configure your browser to work with the Adobe Document Server Web Service administration pages.

Windows installation

Use the following instructions to install and verify Adobe Document Server on Windows.

To install Adobe Document Server on Windows:

- 1 Log on to your system as Administrator.
- 2 Insert the Adobe Document Server CD in the CD-ROM drive and, if necessary, double-click the setup.exe file.
- 3 Enter your user information and the serial number provided with the CD; then click Next.
- 4 Verify your registration information: click Yes if it's accurate, click No and correct it if it is not accurate. Click OK to proceed.
- 5 Read the Software License Agreement carefully. If you accept the terms of the licensing agreement, click Yes. If you do not accept the terms, click No to end the installation.
- 6 Click Next to accept the default installation location, or click Browse to choose a different location.
- 7 In the Custom Setup dialog box select the options you want to install and click Next.
- 8 In the System Variables dialog box, select the options you want to modify your system variables.
- 9 Review your selections and, if necessary, click Back to make changes; otherwise, click Next to complete the installation.

To verify installation:

Do one of the following:

- At the command prompt, type the following command to view the version number, registration information, and the product type (evaluation or full):

```
C:\>altercast -version
```

- Go to <http://localhost:8019/altercast/AlterCast>, and click the GetVersion information link. Click Invoke to view the version number, registration information, and product type. For more information, see the Adobe Document Server documentation.

Solaris installation

Use the following instructions to install, verify, and access samples for Adobe Document Server on Solaris.

To install ADBdocsvr on Solaris:

- 1 Open a command line prompt and verify that you have root access.
- 2 Unzip the Adobe Document Server distribution file, or browse the CD to the location.
- 3 Set your current working directory to the location of the ADBdocsvr file.
- 4 At the command prompt, type the following command exactly as shown; then press Enter:

```
pkgadd -d . ADBdocsvr
```

- 5 Type the serial number provided with the CD and press Enter.
- 6 Read the End User License Agreement (EULA) carefully. To accept the terms of the EULA, type “y” and press Enter.
- 7 Follow the prompts, entering the following information: user name, company name, Tomcat port, and AJP port. (The installer recommends the default ports.)
- 8 Specify an install directory. To install to the default directory, press Enter. To install to another directory, type “n” and press Enter; then type a directory name and press Enter. The installer creates the directory for you.
- 9 To confirm the information entered, type “y” and press Enter. If the information is incorrect, type “n” and enter the correct information.

Note: The installer checks to see if Perl 5 or later is installed. If the correct version of Perl is not installed, you can choose to exit the installation or continue with the installation. If you continue without the correct version of Perl, samples will not be installed.

- 10 When the following message appears, type “y” and press Enter to agree to the creation of the directory:

The selected base directory </opt/adobe> must exist before installation is attempted.

Do you want this directory created now [y,n,?,q]

- 11 When the following message regarding permissions appears, type “y” and press Enter to continue with the installation:

This package contains scripts which will be executed with super-user permission during the process of installing this package. Do you want to continue with the installation of <ADBdocsvr> [y,n,?]

To verify installation:

Do one of the following:

- Open a command window and navigate to the ~/bin directory. At the command prompt, type the following command to view the version number, registration information, and the product type (evaluation or full):
prompt\$ altercast -version
- Go to <http://localhost:8019/altercast/AlterCast>, and click the GetVersion information link. Click Invoke to view the version number, registration information, and product type. For more information, see the Adobe Document Server documentation.

To run the samples install script (so you can access the samples):

- 1 Make sure you are not at the root, and navigate to <ADS installation directory>/samples.
- 2 At the command prompt, type the following, exactly as shown:

```
prompt$ perl install.pl
```

- 3 Select the default installation location, or specify a new location. By default, the samples are installed to your home directory.

Configuring Adobe Document Server for a Web browser

Use the following procedures to configure your internet browser's Proxy settings so that they work with the Adobe Document Server administration pages and Web service pages.

To configure Internet Explorer Proxy settings:

- 1 Open Internet Explorer 5.5 or later.
- 2 Choose Tools > Internet Options.
- 3 Click the Connections tab.
- 4 Click the LAN settings button, and do one of the following:
 - If the Use Automatic Configuration Script option is selected, check with your system administrator to make sure the script properly handles the address you are trying to reach.
 - If the Use Proxy Server option is selected, make sure the Bypass Proxy for Local Addresses is also selected.

To configure Netscape Navigator Proxy settings:

- 1 Open Netscape Navigator 4.7 or later.
- 2 Choose Edit > Preferences.
- 3 Expand the Advanced category, and choose Proxies. If the Automatic Proxy Configuration option is selected, do one of the following:
 - Check with your system administrator to make sure that the script properly handles the address you are trying to reach.
 - Add the name of the machine to the No proxy for section.

Using fonts

Adobe Document Server requires access to all fonts used in the PDF and image files it processes (unless the font is embedded in the PDF document). When you start Adobe Document Server, it searches font directories so that it can accurately create documents and images.

It is important that you set font file and directory permissions properly. Read access may be necessary for all intended users of Adobe Document Server. Directories may require execute permission.

Supported font types for Windows

- PostScript® Type 1 fonts for Windows
- TrueType® fonts for Windows
- OpenType® fonts
- Windows bitmap fonts
- CID-Keyed fonts

Supported font types for Solaris

- PostScript® Type 1 fonts for Windows
- TrueType® fonts for Windows
- OpenType® fonts
- PostScript Type 1 fonts in ASCII format
- CID-Keyed fonts

Locating fonts

Upon startup, Adobe Document Server automatically searches for supported fonts in the following directories:

Windows

- Fonts installed on Windows
- <system drive>:\Program Files\Common Files\Adobe\Fonts
- <program installation directory>/required/fonts

Solaris

- \$OPENWIN/lib/X11/fonts
(using OPENWIN=/usr/openwin, if not defined)
- /usr/lib/X11/fonts
- /usr/local/X11R6/lib/X11/fonts
- <ADS installation directory>/required/fonts

Installing additional fonts

To add fonts to an Adobe Document Server system, do either of the following:

- Install or copy the fonts to the supported font directories (see “Locating fonts” on page 7) and restart Adobe Document Server. When you install the fonts to supported directories, Adobe Document Server finds them automatically.
- Install or copy the fonts to any directory location of your choice. For best performance and reliability, choose a location that is locally mounted (as opposed to a networked location on a remote machine). See the Adobe Document Server documentation for information on how to use the administrative console to tell the Adobe Document Server where these additional directories are located, as well as information on how to use the API methods to instruct the server on where to find the additional font locations.

Note: Adobe recommends that you have a valid license for all of the fonts you use on the Adobe Document Server.

Part 3: Creating documents with XML commands and Adobe Document Server

The command line provides a quick, uncomplicated way for you to invoke Adobe Document Server and examine the results. You can call the Adobe Document Server command line executable, called “altercast”, either manually via a command prompt or from shell scripts/batch files.

The syntax for invoking the command line executable is:

```
prompt$ altercast
or
C:\>altercast
```

Try the following example by copying the XML commands to a new file named “myCommands.xml”. After you enter the commands below into a new file and modify the source paths to point to your own PDF documents, execute the altercast command (as shown above) to create a newly assembled PDF document.

```
<?xml version="1.0" encoding="UTF-8"?>
<commands>
  <loadContent source="pdf1.pdf" out="mypdf1" />
  <loadContent source="pdf2.pdf" out="mypdf2" />
  <assemblePDF>
    <document source="mypdf1" />
    <document source="mypdf2" />
  </assemblePDF>
  <saveContent name="mynewdoc" appendExtension="true" />
</commands>
```

Basic command rules

Adobe Document Server XML commands have a few basic rules to follow. Read these carefully as it will save you time if understand how the commands should be written and in what order the commands are processed.

- Encase all instructions in starting <commands> and ending </commands> wrapper tags.
- Use closing tags (</>) for all empty element tags.
- Make sure attribute values follow the normal XML quoting rules. Adobe recommends enclosing all attribute values in double quotation marks:

```
<loadContent source="pdf1.pdf" out="mypdf1" />
```

- If you are using the command line to invoke Adobe Document Server, specify absolute or relative URIs to the directory in which the commands file resides.
- If desired, enclose values in XPath expressions in single quotation marks:

```
[@name='layername']
```

- Declare the XML version as the first line in XML command files:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Specifying input and output content

Adobe Document Server commands process the current specified content. This means that upon using the `loadContent` command, that file becomes the current content. As you process and make changes to this current content, Adobe Document Server takes the result of one command and uses it as the input for the next command. The current content cascades top to bottom as the data source for each command.

When loading multiple documents, use the “out” attribute within `loadContent` or other XML command to reference the file later. You can then refer this named “out” by using the “in” attribute of any of the XML commands. The “in” attribute is used to reference previously loaded or modified content. If you do not name the output of a command, the resultant content is used only as input to the next command. The named “out” content is also used in other areas of the commands, such as the document tag of the `assemblePDF` command.

Use the “out” and “in” labeling methods only where necessary. Overuse of these attributes results in slower performance each time they are used. Organize your XML to process the commands in the simplest way possible to avoid unnecessary labeling.

Adobe Document Server XML commands

The most commonly used Adobe Document Server commands for dynamic document creation and manipulation are described below.

***Note:** For information on using Adobe Document Server to create images and graphs, see the Adobe Graphics Server Quick Start Guide, which includes an overview of the most commonly used graphics commands. Additional XML commands as well as information on XSL-FO and dynamic document creation using FrameMaker templates are available with the Adobe Document Server documentation.*

<commands></commands> All Adobe Document Server commands must be wrapped in the `<commands>` tag. You can use Adobe Graphics Server commands for the Adobe Document Server.

<assemblePDF /> Allows for customized PDF creation from two or more other PDF documents. The `assemblePDF` command preserves bookmarks, logical structures, and links.

<insertTOC /> Allows you to create and add a dynamic table of contents (TOC) to your document. Use this command after you have used `assemblePDF`. A FrameMaker template is required. The bookmark and page label information in the assembled PDF is examined and applied to the template to form a table of contents.

<overlayTemplates /> Adds dynamic headers and footers or other template styles to your document. A PDF form source document is required.

<overlayPages /> Allows for overlay of one PDF document on top of another; it is similar to the `overlayTemplates` command. Use this command to overlay company logos, stamps (confidential drafts, etc.) and watermarks. Perform all transformations to the PDF overlay before it is applied.

<importFormData /> Allows you to pre-fill information into a PDF form. You can dynamically create XFDF (PDF’s XML Form Data Format) from data obtained from databases or enterprise applications such as SAP, then pass it to Adobe Document Server to be inserted into the document.

<exportFormData /> Extracts a copy of the embedded XFDF data from PDF content and makes it the current content. You can save the XML form data and manipulate it so you can insert it into a database or enterprise application. If you use `exportFormData`, you can use PDF as a container for automating business processes.

<saveContent /> Saves the results of the commands used into a new PDF. Use `saveOptimized` for a Web optimized version of the PDF with a smaller file size.

Using XML command examples

Using a text editor, you can create the example files below and practice executing XML command files using Adobe Document Server command line syntax. Following each example is a detailed, step-by-step explanation of how Adobe Document Server processes your instructions.

Begin by creating a file called “myCommands.xml”. Use the sample files that are included with the installation of Adobe Document Server as the basis for your own commands, or use the samples in combination with your own existing documents that are accessible by the machine that has Adobe Document Server installed.

***Note:** The command line version of Adobe Document Server executes slower than the Web Service or pooled versions that are accessible via the SDKs. This is because for each request made to the command line version, the application must be started up and shut down, whereas for the Web Service and pooled versions, Adobe Document Server is always up, waiting for a request to process.*

Sample PDF files and FrameMaker templates are available in the following locations:

Windows

- <ADS install dir>\samples\PDFManipulation\fmtemplates
- <ADS install dir>\samples\PDFManipulation\test_files
- <ADS install dir>\samples\ImageManipulation\test_files

Solaris

- <ADS samples install dir>/PDFManipulation/fmtemplates
- <ADS samples install dir>/PDFManipulation/test_files
- <ADS samples install dir>/ImageManipulation/test_files

Example 1: Assemble PDF and insert table of contents

This example loads two PDF documents and merges them into a single document. It then applies a table of contents to the file and saves the combined document as a new PDF file.

```
<?xml version="1.0" encoding="UTF-8"?>
<commands>
  <loadContent source="SampleTOC.fm" out="myTOCTemplate" />
  <loadContent source="Chapter1.pdf" out="myp1" />
  <loadContent source="Chapter2.pdf" out="myp2" />
  <assemblePDF>
    <document source="myp1" bookmark="My First Bookmark" />
    <document source="myp2" bookmark="My Second Bookmark" />
  </assemblePDF>
  <insertTOC template="myTOCTemplate" />
  <saveContent name="myDoc" appendExtension="true" />
</commands>
```

To execute your new myCommands.xml file use the following command:

```
c:\>altercast -logLevel 7 -commands myCommands.xml
```

Step-by-step for Example 1:

- 1 Make sure that the first line of your XML file contains the XML version and encoding declaration.

```
<?xml version="1.0" encoding="UTF-8"?>
```

- 2 Wrap the entire contents of the XML file inside the <commands> tag. You can specify attributes to customize your output. The attributes are explained in Example 2.

```
<commands>
...
</commands>
```

- 3 Specify which documents will be used for this process. The loadContent command helps define the files needed. The path to these files may be relative to the myCommands.xml file or be specified by absolute path.

```
<loadContent source="SampleTOC.fm" out="myTOCTemplate" />
<loadContent source="Chapter1.pdf" out="myp1" />
<loadContent source="Chapter2.pdf" out="myp2" />
```

Note: The “out” attribute is defined for each source so that the original file source can be referenced later in the command file.

- 4 Append the PDF documents together to form one document. AssemblePDF instructs Adobe Document Server to add Chapter1.pdf and Chapter2.pdf in that order. (If you need to alter either PDF document, apply those modifications before using the assemblePDF command; then take the resulting content of those changes to merge with the secondary file.)

```
<assemblePDF>
  <document source="myp1" bookmark="My First Bookmark" />
  <document source="myp2" bookmark="My Second Bookmark" />
</assemblePDF>
```

Note: The assemblePDF command can add a new highest level bookmark to each of the PDF files that it is assembling together.

5 Instruct Adobe Document Server to use an earlier input of SampleTOC.fm, which you referenced as “myTOCTemplate”. The FrameMaker template defines paragraph styles for the TOC entries, page numbering, and master pages that control the appearance of the newly generated TOC. When you use the TOC command, the current content’s bookmarks and page label information are used to create the new table of contents. The text listed in the table of contents links to the corresponding page destination. Entries that correspond to the topmost level are the first level of nesting. If your template contains paragraph styles for subsequent nesting levels, these styles will be reflected in your new PDF.

```
<insertTOC template="myTOCTemplate" />
```

6 Save your new PDF using saveContent. Your new file is named myDoc.pdf and will be stored in the same working directory as the myCommands.xml. The appendExtension attribute tells Adobe Document Server to preserve the current file extension to the new named output to create myDoc.pdf.

```
<saveContent name="myDoc" appendExtension="true"/>
```

7 Close the command tag.

```
</commands>
```

Your new file is located at <current directory>/myDoc.pdf

Note: Via command line, you can see step-by-step what Adobe Document Server is processing through server log details. If there are any complications, the new PDF file is not created and an error log is generated.

Example 2: Assemble PDF and overlay templates

This example creates a manual from multiple sources and applies a design template to that document. It loads multiple PDF documents, an image file, and a template file into one output.

This example assumes that you have several of your own PDF files that you want to append together. It also assumes that you have your own FrameMaker template with paragraph styles named TOC-001, TOC-002, etc. Finally, you will need a PDF form that you can use to overlay dynamic headers and footers. If you need help creating each of these components, there are short descriptions below of how to do so.

The following XML is used in this example :

```
<?xml version="1.0" encoding="UTF-8"?>

<commands resultOverwrite="true" resultLocation="file:///c:/pdf/manual">
  <loadContent source="logo.psd" out="logo" />
  <loadContent source="TOCTemplate.fm" out="myTOCTemplate" />
  <loadContent source="HeadersAndFooters.pdf" out="template" />
  <loadContent source="FrontMatter.pdf" out="front" />
  <loadContent source="Section1.pdf" out="one" />
  <loadContent source="Section2.pdf" out="two" />
  <loadContent source="Section3.pdf" out="three" />
  <loadContent source="Other.pdf" out="index" />

  <convertRasterToPDF in="logo" out="logo" />

</assemblePDF>
```

```
<document source="logo" labels="preserve"
bookmark="Introduction" />
  <document source="front" labels="preserve" />
  <document source="one" labels="preserve"
bookmark="Chapter 1" />
  <document source="two" labels="continue"
bookmark="Chapter 2" />
  <document source="three" labels="continue"
bookmark="Chapter 3" />
  <document source="index" labels="continue"
bookmark="Index" />
</assemblePDF>

<setPageLabels start="1" style="decimal" prefix="Page " />

<insertTOC template="myTOCTemplate"/>

  <overlayTemplates pages="2-last" >
    <templateDocument source="template">
<template name="MyEvenPage">
<field name="headerCenter" value="%docInfo.title%" />
<field name="footerLeft"
value="%pageNumber% of %totalPages%"/>
<field name="footerCenter" value="CONFIDENTIAL" />
</template>
<template name="MyOddPage">
<field name="headerCenter" value="%docInfo.title%" />
<field name="footerRight"
value="%pageNumber% of %totalPages%" />
<field name="footerCenter" value="CONFIDENTIAL" />
</template>
</templateDocument>
  </overlayTemplates>

  <updateThumbnails />
  <encryptPDF permissions="print" />

  <saveContent name="manual" appendExtension="true" />

</commands>
```

Step-by-step for Example 2:

- 1 Make sure that the first line of your XML file contains the XML version and encoding declaration.

```
<?xml version="1.0" encoding="UTF-8"?>
```

- 2 Wrap the entire contents of the XML file inside the <commands> tag. In this example, we utilize some of the optional attributes associated with the command tag. The resultOverwrite attribute determines whether to overwrite any existing files modified. A value of true will overwrite changed documents. When the value is set to false, the default, Adobe Document Server may generate an error if it encounters an already existing file.

Using resultLocation changes the default directory to which the new file will be saved. The default is the working directory where the myCommands.xml file is stored. In this case, a complete file:/// URI is specified. File locations are handled differently between the command line version of Adobe Document Server and the Web Service and pooled version you call from the SDKs. From the non-command line version of Adobe Document Server, if you wish to read and write from the file system, you must use an absolute path contained within a file:/// URI. The example below will work both from the command line and from the SDKs.

```
<commands resultOverwrite="true"
  resultLocation="file:///c:/pdf/manual">
```

- 3 Load all the content needed to create your manual. Use loadContent, and be sure to specify the file's reference name using out= "name".

```
<loadContent source="logo.psd" out="logo" />
<loadContent source="TOCTemplate.fm" out="myTOCTemplate" />
<loadContent source="HeadersAndFooters.pdf" out="template" />
<loadContent source="FrontMatter.pdf" out="front" />
```

- 4 Convert the raster Photoshop PSD file to a PDF file. This is necessary because all of the inputs to assemblePDF must be PDF content. To do this, use the convertRasterToPDF command below. Note the use of "in" and "out." You must specify the "in" attribute because you need to use the referenced PSD logo instead of the output from the previous command. You must specify the "out" parameter so that you can reference the new PDF content in the assemblePDF command. Although we use the same name below, it is not required.

```
<convertRasterToPDF in="logo" out="logo" />
```

- 5 As in Example 1, append these files together using assemblePDF. This assemblePDF example also alters page labels in your document. Page labels are located at the bottom of a PDF document in Acrobat Reader. Page information and page number values are contained in the page label.

For instructional purposes, we provide two ways to alter page label information. The first is to use assemblePDF to control your page labels. You can use labels to either use the page label style (roman, decimal, etc.) from the previous document (labels = "continue"), or to retain the label style from the current content (labels = "preserve"). If no option is specified, the default "continue" value is applied.

```
<assemblePDF>
  <document source="logo" labels="preserve"
    bookmark="Introduction" />
  <document source="front" labels="preserve" />
  <document source="one" labels="preserve"
    bookmark="Chapter 1" />
  <document source="two" labels="continue"
    bookmark="Chapter 2" />
  <document source="three" labels="continue"
    bookmark="Chapter 3" />
```

```

    <document source="index" labels="continue"
      bookmark="Index" />
</assemblePDF>

```

The command `assemblePDF` also allows you to create new top-level bookmarks for each of your files. When creating a manual, you can specify a label for major category content sections. The result will be one document with several top-level bookmarks: Introduction, Chapter 1, Chapter 2, Chapter 3, and Index. The bookmarks contained within the assembled PDF documents are preserved also, but can be found one level deep.

```

<document source="logo" labels="preserve"
  bookmark="Introduction" />
<document source="one" labels="preserve"
  bookmark="Chapter 1" />
<document source="two" labels="continue"
  bookmark="Chapter 2" />
<document source="three" labels="continue"
  bookmark="Chapter 3" />
<document source="index" labels="continue"
  bookmark="Index" />

```

The next line also instructs Adobe Document Server to add page labels to the specified pages (this line overwrites your previous page label conditions). You can preserve the page labels for the logo page, TOC, and front page and begin the new page label on page 3. The decimal style preceded by “Page” will be applied for these subsequent pages through the end of the document.

```

<setPageLabels start="3" style="decimal" prefix="Page " />

```

6 Insert a table of contents with the `insertTOC` command. By positioning this command near the end of your document creation, the newly assembled PDF will be applied to the generated table of contents.

```

<insertTOC template="myTOCTemplate"/>>

```

The `insertTOC` command works by applying paragraph styles defined in the FrameMaker template to the bookmark titles contained within the PDF files. The paragraph styles are applied to the content and placed on the master page. The paragraph styles must be named `TOC-###`. For example, for the first level bookmark, Adobe Document Server applies the paragraph style named `TOC-001` to the text. For the second level bookmark, `TOC-002` is applied. (As mentioned earlier in the example, you should create your FrameMaker template inside of Adobe FrameMaker. The bookmark structure of the PDF document is used to create the table of contents.)

You can also create different master pages for the first page, even pages, and odd pages. (See the Adobe FrameMaker documentation if you need help with creating paragraph styles or master pages. Adobe also recommends the Classroom in a Book series of books if you need additional instruction in FrameMaker.)

Adobe Document Server creates a table of contents with the appropriate styles on the appropriate pages. If you have only defined two `TOC-###` paragraph styles, Adobe Document Server places the top two bookmark levels in the generated table of contents.

7 Apply a template style to the current content. Using a PDF template file, you can specify template conditions to certain pages within the document. The template file must be a PDF form. In the example below, a series of dynamic headers and footers are applied to the document. Begin these changes at any point in the document. If you prefer not to add a template style to the opening page, begin `overlayTemplates` on the pages that contain main content as shown in this example.

```
<overlayTemplates pages="2-last" >
```

Note: The PDF file needed for `overlayTemplates` must be a PDF form. Without a PDF file containing form fields, Adobe Document Server will not complete your request. The line below references the content named `template` that we named in a previous `loadContent`.

```
<templateDocument source="template">
```

The command superimposes all text, graphics, and image elements from the template file onto the current content. It populates the form fields in the template, then flattens all the form fields that occur in the template and superimposes them onto your document. Form fields can be populated with dynamic content from your template. For fields specified by a `<field>` element, this command replaces the value as instructed.

To create a PDF form, you need the full version of Adobe Acrobat. Add form fields to a static PDF document. The names you give to the form fields inside of Acrobat are the same names that you specify in the field name attribute. In the example below, Adobe Document Server inserts the text “CONFIDENTIAL” into the field named `footerCenter`.

```
<field name="footerCenter" value="CONFIDENTIAL" />
```

8 After you have added form fields to your PDF template, create page templates for each of the pages you want to overlay onto your document. To create page templates in Acrobat, choose `Tools > Forms > Page Templates`. The names you give to each page are the names that are referenced in the `need to add form fields to a static PDF document`. The names you give to the name attribute of the template tag. Below, the page template named `MyEvenPage` is applied, followed by `MyOddPage`. After `MyOddPage` has been applied, `MyEvenPage` is applied, and so on.

```
<template name="MyEvenPage">
  <field name="headerCenter" value="%docInfo.title%" />
  <field name="footerLeft"
value="%pageNumber% of %totalPages%"/>
  <field name="footerCenter"
value="CONFIDENTIAL" />
</template>
<template name="MyOddPage">
  <field name="headerCenter" value="%docInfo.title%" />
  <field name="footerLeft"
value="%pageNumber% of %totalPages%"/>
  <field name="footerCenter"
value="CONFIDENTIAL" />
</template>
```

Adobe Document Server understands certain variables that you can use for processing the overlayTemplate request. These variables can be called under this command tag and are designated with the % sign. (For a complete reference of supported variables, see the documentation that shipped with Adobe Document Server. For additional information on creating PDF forms, see the Acrobat online Help and the Forms chapter of the Acrobat Classroom in a Book.)

```
... value="%pageNumber% ...
```

9 Ensure that the embedded thumbnails reflect the changes made to the new document by using the updateThumbnails command. Update thumbnails for the entire document. Page ranges for updating thumbnails can be specified but not used for this example. See the product documentation for more information.

```
<updateThumbnails />
```

10 Activate the Adobe Standard Security Handler by using the encryptPDF command for Adobe Document Server. This simple XML command secures your document with password protection, permissions and encryption options. In this example, you will control the permission to view the file of any user for this document regardless if the user has Acrobat Reader or a full version of Acrobat. The new permission is read-only by setting the value of the permission to “print”. Note that when not specified, permission for all documents default to “full access.” Use this feature to prohibit others from editing or selecting text information from your PDF document.

```
<encryptPDF permissions="print" />
```

11 Save the output to a file named manual.pdf.

```
<saveContent name="manual" appendExtension="true" />
```

12 Close the command tag.

```
</commands>
```

Your new file is located at c:\pdf\manual\manual.pdf.

Note: Via command line, you can see step-by-step what Adobe Document Server is processing through server log details. If there are any complications, the file will not be created and an error log will be generated.

Part 4: Communicating to Adobe Document Server with an API

Adobe Document Server integrates with industry standard APIs—Java, Perl, COM, and SOAP. Each API is capable of sending requests to Adobe Document Server. This document does not include information on SOAP, though you can use the Web Service version of Adobe Document Server through the simplified Java and Perl APIs.

How the SDKs work

Each SDK requires the creation of three objects: Server, Request, and Response. Following is a simple breakdown of how the SDK creates and sends requests to Adobe Document Server:

- 1 Create a *Server* Object, which instantiates Adobe Graphics Server.
- 2 Create a *Request* Object of XML commands. Pass the request to the server for execution.
- 3 Get a *Response* and process it further (save to disk, a content management system, return it as an HTTP response in your application).

Examples of Java, Perl, and COM APIs

In each of the following API examples, there are two ways of having Adobe Document Server load data. The first is to specify an absolute file:/// URI to load content. In this case, the file content must be accessible to the machine that is executing the server. If the file is not accessible, you need to use a different form of loadContent. Specify a name in the source="" attribute, and then use the request.addFile method to add that file to the request. The file will be loaded by the SDK on the client and sent to the server.

The code used in these examples is a command line Java, Perl, or COM application that uses the associated SDK to execute the commands. When developing your application to integrate with the SDK, the XML commands would not be hard coded as seen in the example code, but would most likely be built by your application, based on a set of input parameters.

Using the examples, enter the code into a text editor; then compile and run it. Feel free to experiment with the code or use a different XML command example.

(For more information, see the complete SDK reference documentation that was included with Adobe Document Server. The documentation also includes other examples of how to use the Java, Perl, or COM API to call Adobe Graphics Server to create dynamic graphics.)

Example of Java API

The Java SDK interface for invoking Adobe Document Server is available for local and remote execution through the Web Service on Windows and Solaris. You can use the Java SDK in any environment that supports Java, such as a Java Web Application Server, a Java application, servlets, or Java Server Pages (JSP).

```
import com.adobe.altercast.sdk.*;

import java.io.*;
import java.net.*;
import java.util.*;

public class test {

    public static void main(String [] args) {

StringBuffer cmd = new StringBuffer(1024);
```

```
// Create the XML Commands
// -----
// This example loads multiple PDF documents, and assembles them
// into one output PDF. It also creates a table of contents.

cmd.append("<?xml version=\"1.0\" encoding=\"UTF-8\"?\>\n\n");

// To have the Document Server write the files directly to disk,
// uncomment the line below. The active line causes the files
// to be returned through the SDK for you to process further.

//cmd.append("<commands resultOverwrite=\">true\" resultLocation=\"file:///export/home/development/ads/manual\">\n";
cmd.append("<commands>\n");

// Load Content
// -----
// Tell Adobe Document Server to load the files from disk
// Specify the full path to these files
// Use file:/// URI syntax

cmd.append("<loadContent source=\"file:///export/home/development/ads/SampleTOC.fm\"
out=\"myTOCTemplate\" />\n");
cmd.append("<loadContent source=\"file:///export/home/development/ads/Chapter1.pdf\" out=\"myp1\"
/>\n");
cmd.append("<loadContent source=\"file:///export/home/development/ads/Chapter2.pdf\" out=\"myp2\"
/>\n");

cmd.append("<assemblePDF> \n");
cmd.append(" <document source=\"myp1\" bookmark=\"My First Bookmark\" /> \n");
cmd.append(" <document source=\"myp2\" bookmark=\"My Second Bookmark\" /> \n");
cmd.append("</assemblePDF> \n");

cmd.append("<insertTOC template=\"myTOCTemplate\"/>\n");
cmd.append("<saveContent name=\"myDoc\" appendExtension=\"true\" />\n");
cmd.append("</commands>\n");

// Create Server Object
// -----
// Create a reference to web service server. If you create a
// direct pooled server, you must manually manage that pool.
// Adobe recommends using the web service server.

Server docServer = null;
String docServerLocation = "http://localhost:8019/altercast/AlterCast";

try {
    docServer = AlterCast.createWebServiceServer(new URL(docServerLocation));
```

```
} catch (Exception e) {

    System.err.println("Could not find the specified Server at");
    System.err.println(docServerLocation);
    System.err.println("Details follow:");
    e.printStackTrace(System.err);

}

try {

// Create a Request
// -----
// Create a request to the server

    Request myRequest = AlterCast.createRequest();

// Add to the Request
// -----
// Add your XML Commands to the request

    myRequest.setCommands(cmd.toString());

// Execute the Request
// -----
// Instruct the server to process your request

    Response myResponse = docServer.execute(myRequest);

// Get the results
// -----
// Extract the results from the response

    ResultContent[] result = myResponse.getResults();

// Write the results to disk
// -----

    for (int i=0; i < result.length; i++) {

        ResultContent myresult = result[i];

        File myFile = new File("/export/home/development/ads/", myresult.getName());

        byte[] resultBody = myresult.getData();
        FileOutputStream stream = null;

    try {

        stream = new FileOutputStream(myFile);
```

```
try {

stream.write(resultBody);

} catch (Exception e) {

System.err.println("Could not write to file: " + myFile.getAbsolutePath());
System.err.println("Java err: " + e);

} finally {

if (stream != null) stream.close();

}
} catch (Exception e) {

System.err.println("Could not create output stream for file: " + myFile.getAbsolutePath());
System.err.println("Java err: " + e);

}

}

} catch (Exception e) {

    System.err.println("Program failed.");
    System.err.println("Details follow:");
    e.printStackTrace(System.err);

} finally {

// Release Adobe Document Server
// -----
// You should always release a server when finished.
// This step is critical in releasing active requests.
// Using this command releases those processes.
// Without this method, the processes in a pool will
// not be made available to other threads that need it.

try {
    docServer.release();

} catch (Exception e) {

    ; // do nothing

}

}

}

}
```

Example of Perl API

The Perl SDK interface for invoking Adobe Document Server is available for local and remote execution through the Web service on Windows and Solaris. You can use the Perl SDK in any environment that supports Perl.

```
#!/usr/bin/perl
# *****
# test.pl
#
# This example loads multiple PDF documents, and assembles them
# into one output PDF. It also creates a table of contents.
#
# *****

#
# Figure out where the SDK is and include the various directories you
# need to actually find your libraries.

BEGIN
{
    my $pathToSDK = "/opt/adobe/documentserver.5.0/sdk/perl/src/engine";
    unshift (@INC, $pathToSDK);
}

use strict;
use FileHandle;
use AlterCast;

#
# Generate the commands that we will use to perform
# the text replacement.

my $commands = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n\n";

# To have the Adobe Document Server write the files directly to disk,
# uncomment the line below. The active line causes the files
# to be returned through the SDK for you to process further.

#$commands = "<commands resultOverwrite=\"true\" resultLocation=\"file:///export/home/development/ads/manual\">\n" .
$commands = $commands . "<commands>\n" .
    "\t<loadContent source=\"file:///export/home/development/ads/SampleTOC.fm\" out=\"myTOCTemplate\"
/> \n" .
    "\t<loadContent source=\"file:///export/home/development/ads/Chapter1.pdf\" out=\"myp1\" /> \n" .
    "\t<loadContent source=\"file:///export/home/development/ads/Chapter2.pdf\" out=\"myp2\" /> \n" .
    "\t<assemblePDF>\n" .
    "\t\t<document source=\"myp1\" bookmark=\"My First Bookmark\" />\n" .
    "\t\t<document source=\"myp2\" bookmark=\"My Second Bookmark\" />\n" .
    "\t</assemblePDF>\n" .
    "\t<insertTOC template=\"myTOCTemplate\" /> \n" .
    "\t<saveContent name=\"myDoc\" appendExtension=\"true\" /> \n" .
"</commands>\n";
```

```
# Load the default server location
my $webServiceLocation = "http://localhost:8019/altercast/AlterCast";

# Create the server
my $server = AlterCast->createWebServiceServer(URL => $webServiceLocation);

if (!defined $server) {
    print STDERR "An error occurred while creating the server:\n";
    print STDERR AlterCastErrors->getLastError();
}

my $request;
$request = AlterCast->createRequest();

# add the XML commands to the request
$request->setCommands(Commands => $commands);

# Send the request and get response.
my $response = $server->execute(Request => $request);

if (!defined $response) {
    print "An error occurred while executing the request:\n";
    print AlterCastErrors->getLastError();
    exit;
}

# get the results from the server
my @results = $response->getResults();
my $firstResult = $results[0];

# Write the content out to our result file.
# it will be written in the current directory
my $outputfile = $firstResult->getName();
my $output_fh = new FileHandle();
open $output_fh, ">$outputfile";

if ($output_fh) {
    binmode $output_fh;
    print $output_fh $firstResult->getData();
    close $output_fh;
}
else {
    print STDERR "Could not open output file $outputfile\n";
}

# The commands are done, so we can now clean up
$server->release();
```

Example of COM API

The COM SDK interface for invoking Adobe Document Server is available for local and remote execution through the mechanisms provided by COM+ on Windows only. You can use the COM SDK in any environment that supports it, including any of these languages: JScript, VBScript, Visual Basic, .NET, VB.NET, C, or C++.

If you are using .NET, you can access Adobe Document Server either through the COM object or by setting a “web reference” to the Web Service address (“http://localhost:8019/altercast/AlterCast”), and then using the automatically created classes from within C#.

```
<%@ language=javascript %>

<%
/* *****
*
* test.asp
*
* This example loads multiple PDF documents, and assembles them
* into one output PDF. It also creates a table of contents.
*
* *****
%>

<%
    // Create the commands xml
    //
    var cmd = "<?xml version='1.0' encoding='UTF-8'?">";

    // To have the Document Server write the files directly to disk,
    // uncomment the line below. The active line causes the files
    // to be returned through the SDK for you to process further.

    //cmd = cmd + "<commands resultOverwrite='true'"
resultLocation="file:///c:/pdf/manual">\n";
    cmd = cmd + "<commands>\n";
    cmd = cmd + "\t<loadContent source='file:///c:/pdf/SampleTOC.fm'"
out="myTOCTemplate"/>\n";
    cmd = cmd + "\t<loadContent source='file:///c:/pdf/Chapter1.pdf' out='myp1'/">\n";
    cmd = cmd + "\t<loadContent source='file:///c:/pdf/Chapter2.pdf' out='myp2'/">\n";
    cmd = cmd + "\n";
    cmd = cmd + "\t<assemblePDF>\n";
    cmd = cmd + "\t\t<document source='myp1' bookmark='My First Bookmark' />\n";
    cmd = cmd + "\t\t<document source='myp2' bookmark='My Second Bookmark' />\n";
    cmd = cmd + "\t</assemblePDF>\n";
    cmd = cmd + "\n";

    cmd = cmd + "\t<insertTOC template='myTOCTemplate'/">\n";
    cmd = cmd + "\t<saveContent name='myDoc' appendExtension='true' />\n";
    cmd = cmd + "</commands>\n";

    //
```

```
// Create the server object

var theServer = Server.CreateObject("AlterCastCOM.ACServer");

//
// Create the request

var theRequest = Server.CreateObject("AlterCastCOM.ACRequest");

//
// Add the XML Commands to the request.

theRequest.SetCommands(cmd);

//
// Execute the request and get the response.

var theResponse = theServer.Execute(theRequest);

//
// Get the contents from the Document Server, then write them out
// as the response

var theData = theResponse.getData();
var theRecord = theData.Item(1);

Response.ContentType = theRecord.getType();
Response.BinaryWrite(theRecord.getData());

//
// Clean up.

theRecord = null;
theData = null;
theResponse = null;
theServer = null;
theRequest = null;

%>
```