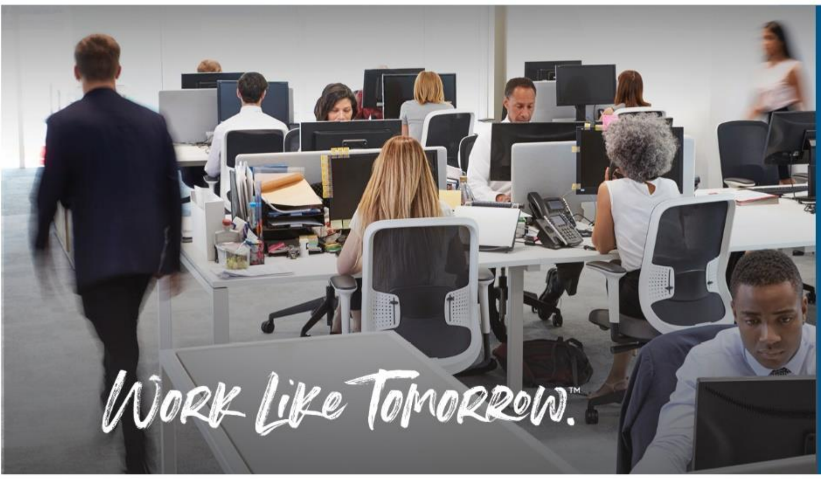

Kofax OmniPage Server Sample Reference Guide

Contents

Introduction

The OmniPage Server package includes sample applications, written in C# and Java that demonstrate the main features of the OmniPage Server API and the Service Data Objects.

The sample applications do not include exception handling input validation and do not use threading to increase performance. Use the applications only as examples but do not use them without modification in the production environment.

The C# samples include C# projects for Microsoft's Visual Studio 2013.

Samples

Helper library for C# Sample Applications

Most of the samples are based on the Kofax.OmniPage.Server.ClientBase library. This library is wrapped around the client – server communications and offers an interface very similar to the OmniPage Server service interface.

Solution	Project	Description
ClientBase	Kofax.OmniPage.Server.ClientBase	OmniPage Server communication and service data objects assembly.

Console Samples

Basic Sample

This absolute minimalistic example is implemented in the Kofax.OmniPage.Server.ClientBase assembly. It creates a job, uploads one input document to convert and download the result document. All the parameters should be entered in the source code, so this sample does not perform command line processing, the conversion is performed with default parameters.

The example performs the following steps:

1. Initializes the ClientBase wrapper in the background which sets the user credentials.
2. Creates a job with the given job type.
3. Gets the upload URL for the input file and uploads it to the server.
4. Starts the conversion with default parameters.
5. Polls the job state information until it is completed.
6. Gets the download URL of the result file and downloads it.

Simple Job

This sample performs a simple conversion in the easiest way. It is implemented in the Kofax.OmniPage.Server.SimpleJobSample assembly. This sample creates a job, uploads one input document to convert and downloads the result document. The parameters can be passed through command line argument to the sample application.

The sample performs the following steps:

1. Processes its command line arguments to fill the required parameters.
2. Initializes the ClientBase wrapper in the background which sets the user credentials.
3. Gets the available job types from the server. This call demonstrates how to get job types, but it is unnecessary for the job itself.
4. Creates a job with the given job type.
5. Gets the upload URL for the input file and uploads it to the server.
6. Starts the conversion with the given conversion parameters.
7. Polls the job state information until it is completed.
8. Gets the download URL of the result file and downloads it.

Required parameters

Parameter	Description
ConversionServiceEndpoint	The base URL of the conversion service. Set it in app.config file.
Input	The path of the input document to convert.
Output	The path of the result document.
Type	The job type identifier.

Sample command line

```
SimpleJobSamples.exe -input=c:\input\test.jpg -output=c:\output\test.docx -type=13
```

Convert from FileShare

This example does not upload the input file to the server, only sets the UNC path to the job where the input files can be found. This kind of job can work only if the server can access the input files on the specified UNC path.

This pattern is the most efficient use of the server, because the file transfer does not use the WEB API, the copying of the input/output files are kept minimal and the job can be created and started with one API call.

The easiest way to use this pattern is to make a symbolic link to the file share where the input files can be found. You must create a link with the same name on all Worker machines, for example mklink /d c:\FileShare \\MyFileShare\OPS\.

Note: It is important that the 'service user' who runs the service has permission to read the input files and to create a new folder next to the input files to store the result files.

The result files are stored automatically on the file share next to the input files, in a folder which has a name matching the identifier of the job.

The client performs the following steps:

1. Processes its command line arguments to fill the required parameters.
2. Initializes the ClientBase wrapper in the background which sets the user credentials.
3. Creates and starts a job in one step with the given job type and input file.
4. Gets the status of the job.

Required parameters

Parameter	Description
ConversionServiceEndpoint	The base URL of the conversion service. Set it in app.config file.
input	The URI of the input document to convert.
type	The job type identifier.

Sample command line

ConvertFromFileShare.exe -input=c:\FileShare\input\test.jpg -type=13

Workflow Sample

This sample requires a workflow xml document which can be created with the Workflow Designer tool of Kofax.

The 'Workflow Sample' performs a workflow-based job on the OmniPage Server. There are two types of input files:

- the image files to be recognized,
- the workflow xml file which describe the steps of the recognition process and its parameters.

The test application performs the following steps:

1. Processes its command line arguments to fill in the required parameters.
2. Initializes the ClientBase wrapper in the background which sets the user credentials.
3. Creates a job with the workflow job type (id = 200).
4. Gets the upload URLs for the input files and uploads them to the server.
5. Creates JobDataDescription to identify the workflow xml file.

6. Starts the conversion with the given conversion parameters.
7. Gets the job state information until it is completed.
8. Gets the download URL of the result file and downloads it.
9. Gets the JobDataDescription to identify the output files and the result xml files.

Sample command line

```
WorkflowSample.exe  -input=c:\input\test.jpg -workflow_xml=c:\input\workflow.xml
  -output=c:\output\result.txt
```

Form Processing Sample

This sample performs a form recognition job. There are two types of input files:

- the image files to be recognized (filled forms)
- the form template library to fit into the forms.

The result XML file contains the data extracted from the filled forms.

The form recognition based on the form template library which can be created by the Form Template Editor by Kofax.

The test application performs the following steps:

1. Processes its command line arguments to fill the required parameters.
2. Initializes the ClientBase wrapper in the background which sets the user credentials.
3. Creates a job with the job type 29 (a form recognition job).
4. Gets the upload URLs for the input files and uploads them to the server.
5. Creates JobDataDescription to describe which files are input images and which is form template.
6. Starts the conversion with the given conversion parameters.
7. Gets the job state information until it is completed.
8. Gets the download URL of the result file and downloads it.

Required parameters

Parameter	Description
ConversionServiceEndpoint	The base URL of the conversion service. Set it in app.config file.
input	The path of the input document to convert.
workflow_xml	The path of the workflow xml file.
Output	The path of the result document.

Sample command line

```
FormProcessingSample.exe  -input=c:\input\form.tif -formtemplate=c:\input\template.ftl
  -output=c:\output\form_data.xml
```

Document Classification

This example Document Classification job requires a ‘knowledge base’ file which can be created with the export function of the ‘Document Classifier Assistant tool.

The sample performs a document classification on the OmniPage Server. There are two types of input files:

- the image files that needs classification
- the ‘knowledge base’ file which describes the classification process.

The test application performs the following steps:

1. Processes its command line arguments to fill the required parameters.
2. Initializes the ClientBase wrapper in the background which sets the user credentials.
3. Creates a job with the document classification job type (id = 36).
4. Gets the upload URLs for the input files and uploads them to the server.
5. Creates JobDataDescription to describe which file is the ‘knowledge base’ file.
6. Starts the job with the given conversion parameters.
7. Gets the job state information until it is completed.
8. Gets the download URL of the result file and downloads it.
9. The result xml document contains the detailed description of the classification result in xml format.

Required parameters

Parameter	Description
ConversionServiceEndpoint	The base URL of the conversion service. Set it in app.config file.
input	The path of the input document to convert.
dcproject	The path of the Document Classifier project file
output	The path of the classification result file.

Sample command line

```
DocumentClassificationSample.exe -input=c:\input\image.jog -dcproject=c:\input\classifier.dcp
  -output=c:\output\classifier_result.xml
```

Advanced Samples

Document Tunneling Sample

These samples are GUI based applications written in .NET Windows Presentation Foundation.

In this sample the user can select multiple output formats and the sample code creates a separated job for all selected output types.

The client downloads the available job types (output formats) from the server at startup. Then the input files, the output folder and the conversion parameters can be defined.

The jobs are running parallel and when one job is completed its result files are downloaded into the output folder.

Parameter	Description
ConversionServiceEndpoint	The base URL of the conversion service. Set it in app.config file.
Input Files	The path of the input documents to convert. (GUI)
Output Folder	The output folder path. (GUI)
Output format(s)	The job types. (GUI)

Notification Sample

This sample demonstrates how to implement your own notification service and how to configure a job to use this service.

It contains two solutions: the *NotificationService* and the *NotificationClient*.

Notification Service

This project implements a web service that corresponds to the Notification Service specification of the OmniPage Server. For details, see the *API Reference Guide*.

The project is configured to use IIS to host the web application, so the Internet Information Services Windows component must be turned on.

Configure the following parameters in the web.config file.

Parameter	Description
ConversionServiceEndpoint	The base URL of the conversion service.

LocalStoragePath	Path to a folder where the user (ASP.NET or Network Service, depending on the IIS version) has full control. The local storage service stores the result files here.

Your notification service must be available to the OmniPage Server through http(s), because it tries to call the Notification service when the job's state changes.

On the NotifyJobCompletion call, the service downloads the results of the given job from the server and stores it in the local file system.

When the NotifyJobFailure is called, the job information and the detailed description of the failure are saved in the error log.

Notification Client

This sample application creates a job, and instructs the OmniPage Server to send notification about the status if a job state changes to Completed or Failed.

This sample is based on the Simple Job Sample, but the job notification is additionally configured to use your own notification service.

Parameter	Description
ConversionServiceEndpoint	The base URL of the conversion service. Set it in app.config file.
NotificationServiceEndpoint	The base URL of your notification service. Set it in app.config file.
input	The path of the input document to convert.
type	The job type identifier.

Java Samples

The Java samples here contain NetBeans (8.2) projects based on JDK 1.8.

Configure java environment

To build the NetBeans/Java environment, perform the following steps:

Installations

1. Install Java SDK at least JDK 1.8
2. Install the NetBeans IDE

3. Unpack the java samples.
4. Start the NetBeans IDE and create a new Project Group.
    a. Select the File/Project Groups…/New Group …
    b. Choose the “Folder of Projects” and browse the folder which contains the java samples.
5. Clean and build the ClientBase project first, then all the other projects.

Helper library for Java Sample Applications

Like the .NET, Java samples use a helper library called “ClientBase”.

ClientBase is a higher-level abstraction built on top of java.net.HttpURLConnection to simplify the most common client server communication. This library also contains service objects to make configuration simpler.

BasicSample

This is the simplest java sample. It uses the Kofax.OmniPage.Server.ClientBase library to perform the client-server communication. This sample creates a simple OPS job, uploads one input document to convert and downloads the result document.

The client performs the following steps:

1. Processes its command line arguments to fill the required parameters.
2. Initializes the ClientBase class in the background.
3. Creates a job with the given job type.
4. Gets the upload URLs for the input files and uploads them to the server.
5. Starts the conversion with the given conversion parameters.
6. Polls the job state information until it is completed.
7. Gets the download URL of the result file and downloads it.

Required command line parameters

Parameter	Description
-s,--service_url	The base URL of the conversion service.
-i,--input_files	Comma separated list of input document’s path.
-o,--output_file	Output file path.
-j,--job_type_id	The job type identifier.

Sample command line

java -jar BasicSample.jar –s http:/my_ops_server/OmniPage.Server.Service -i c:\work\test.jpg -o c:\work\test.txt -j 6

Convert from FileShare

This example does not upload the input file to the server but sets only the UNC path to the job where the input files are located. This kind of job can work only if the server can access the input files on the specified UNC path.

This pattern is the most efficient use of the server because the file transfer does not use the WEB API, the copying of the input/output files are kept minimal and the job can be created and started by one API call.

The easiest way to use this pattern is to make a symbolic link to the file share where the input files are located. You must create a link with the same name on all Worker machines, for example

mklink /d c:\FileShare \\MyFileShare\OPS\.

Note: It is important that the 'service user' on whose behalf the service runs, has permission to read the input files and create a new folder next to the input files to store the result files.

The result files are stored automatically on the file share next to the input files in a folder whose name matches the job identifier.

The client performs the following steps:

1. Processes its command line arguments to set the required parameters.
2. Initializes the ClientBase class in the background.
3. Creates and starts a job in one step with the given job type and input files.
4. Polls the status of job.
5. Gets the location of the result files.

Required parameters

Parameter	Description
-s,--service_url	The base URL of the conversion service.
-i,--input_files	Comma separated list of the UNC path of the input document.
-j,--job_type_id	The job type identifier.

Sample command line

```
java -jar FileShareSample.jar -s http://my_ops_server/OmniPage.Server.Service -i
\\share\test.jpg -j 6
```

Form Processing Sample

This sample performs a form data extraction job. There are two types of input files:

- the image files to be recognized (filled forms)
- the form template library to fit into the forms.

The result file contains the data extracted from the filled forms in an XML document.

The form recognition based on the form template library which can be created by the Form Template Editor of Kofax.

The test application performs the following steps:

1. Processes its command line arguments to fill the required parameters.
2. Initializes the ClientBase class.
3. Creates a job with the job type 29 (a form recognition job).
4. Gets the upload URLs for the input files and uploads them to the server.
5. Creates JobDataDescription object to describe which files are input images and which are form templates.
6. Starts the conversion with the given conversion parameters.
7. Polls the job state information until it is completed.
8. Gets the download URL of the result file and downloads it.

Required parameters

Parameter	Description
-s,--service_url	The base URL of the conversion service.
-i,--input_files	Comma separated list of the path of input documents.
-tl,--template_library	Form template library path.
-o,--output_file	Output file path.

Sample command line

```
java -jar FormProcessingSample.jar -s http:/my_ops_server/OmniPage.Server.Service -i
c:\work\form1.tif,c:\work\form2.tif -tl c:\work\form_template_library.ftl -o c:\work\result.xml
```