

Migrating From PDFReactor 8.0 to PDFReactor 8.1

JAVA AND .NET WRAPPERS

When using either the Java or .NET wrapper APIs, your integration has to be adjusted. These wrapper APIs are now based on the REST API rather than the SOAP API. This was done to provide an API that is more in line with the other wrapper APIs.

PDFREACTOR WEB SERVICE SETTINGS

When using the PDFReactor Web Service with custom settings in the "pdfreactorwebservice.voptions" file, you have to migrate these settings to the "start.ini" located in the "PDFReactor/jetty" directory.

If your "pdfreactorwebservice.voptions" looked like this

```
-Xmx1024m  
-Djava.awt.headless=true
```

you have to add the lines from your "pdfreactorwebservice.voptions" to the end of your "start.ini" file

```
--exec  
-Dorg.apache.cxf.Logger=org.apache.cxf.common.logging.Slf4jLogger  
-Dcom.realobjects.interceptConsoleOutput=true  
# old pdfreactorwebservice.voptions settings  
-Djava.awt.headless=true  
-Xmx1024m
```

PDFreactor Migration Guide

Migrating From PDFreactor 7- to PDFreactor 8+

With PDFreactor 8 we are introducing the first major API change since PDFreactor 2. One major benefit of this change is that the new Java API is identical (with a few additions) to the newly introduced Java web service client API.

USING THE LEGACY JAVA API

Java integrators can still use the old API, however we highly recommend to migrate to the new API as soon as possible, since the old one will be removed in a future release of PDFreactor. To continue using the old legacy API, just change the package from

```
com.realobjects.pdfreactor
```

to

```
com.realobjects.pdfreactor.legacy
```

MIGRATING TO THE NEW API

The Configuration Object

The most obvious change is the introduction of the `Configuration` object. In previous versions of PDFreactor, you invoked all API methods on the PDFreactor instance, however this had some disadvantages like making the PDFreactor instance non-reusable. In PDFreactor 8, you just have to create one instance of PDFreactor. The instance only has a few API methods, like `convert`.

All settings and options are properties of the configuration and have to be set there. While most methods are the same as in previous versions of PDFreactor, some have changed. For example, instead of add-methods, getters are used to retrieve lists on which new entries can be added. Make sure to consult the API documentation.

PDFreactor Migration Guide

Converting

To create a PDF or image, you now just have to call the `convert` method with the configuration as a single parameter, which also specifies the input document. PDFreactor automatically detects if you are converting an HTML string, a URL or binary data.

Retrieving the Result

The new `convert(Configuration)` method no longer returns the PDF as binary directly. It returns a `Result` object which not only contains the PDF as binary, but also other useful data such as the log.

There are additional `convert` methods, such as `convert(Configuration, OutputStream)` which writes the PDF directly in the specified `OutputStream` instead of returning it or `convertAsBinary(Configuration)` which returns the binary data directly instead of a `Result` object. Please make sure to read the API documentation and the PDFreactor manual (Chapter "Integration").

EXAMPLES

Below are simple examples in different programming languages that show how PDFreactor was used previously and how it is used now.

- [Java \(page 4\)](#)
- [.NET \(page 5\)](#)
- [PHP \(page 6\)](#)
- [Python \(page 7\)](#)
- [Ruby \(page 8\)](#)
- [Perl \(page 9\)](#)

PDFReactor Migration Guide

Java

Old API

```
PDFReactor pdfReactor = new PDFReactor();

// simple settings
pdfReactor.setAddBookmarks(true);
pdfReactor.setAddLinks(true);

// adding a user style sheet
pdfReactor.addUserStyleSheet("p { color: red }", null, null, null);

// create PDF and specify the document
byte[] pdf = pdfReactor.renderDocumentFromURL("http://www.realobjects.com");
```

New API

```
PDFReactor pdfReactor = new PDFReactor();
Configuration config = new Configuration();

// specify the document
config.setDocument("http://www.realobjects.com");

// simple settings
config.setAddBookmarks(true);
config.setAddLinks(true);

// adding a user style sheet
config.getUserStyleSheets().add(new Resource("p { color: red }", null));

// create PDF
Result result = pdfReactor.convert(config);
byte[] pdf = result.getDocument();
```

PDFreactor Migration Guide

.NET

Old API

```
PDFreactor pdfReactor = new PDFreactor();

// simple settings
pdfReactor.SetAddBookmarks(true);
pdfReactor.SetAddLinks(true);

// adding a user style sheet
pdfReactor.AddUserStyleSheet("p { color: red }", "", "", "");

// create PDF and specify the document
byte[] pdf = pdfReactor.RenderDocumentFromURL("http://www.realobjects.com");
```

New API *(Changed in version 8.1)*

```
PDFreactor pdfReactor = new PDFreactor();
Configuration config = Configuration();

// specify the document
config.Document = "http://www.realobjects.com";

// simple settings
config.AddBookmarks = true;
config.AddLinks = true;

// adding a user style sheet
config.UserStyleSheets = new List<Resource> {new Resource("p { color: red }", "")};

// create PDF
Result result = pdfReactor.Convert(config);
byte[] pdf = result.Document;
```

PDFReactor Migration Guide

PHP

Old API

```
$pdfReactor = new PDFReactor();

// simple settings
$pdfReactor->setAddBookmarks(true);
$pdfReactor->setAddLinks(true);

// adding a user style sheet
$pdfReactor->addUserStyleSheet("p { color: red }", "", "", "");

// create PDF and specify the document
$result = $pdfReactor->renderDocumentFromURL("http://www.realobjects.com");
```

New API

```
$pdfReactor = new PDFReactor();
$config = array(
    // specify the document
    "document" => "http://www.realobjects.com",

    // simple settings
    "addBookmarks" => true,
    "addLinks" => true,

    // adding a user style sheet
    "userStyleSheets" => array(
        array(
            "content"=> "p { color: red }"
        )
    )
);

// create PDF
// ...as base64 encoded String
$result = $pdfReactor->convert($config);
$pdf = $result->document;

// ...as binary
$pdf = $pdfReactor->convertAsBinary($config);
```

Note: To convert the base64 encoded document into binary data, you can do the following:

```
echo base64_decode($result->document);
```

PDFreactor Migration Guide

Python

Old API

```
pdfReactor = PDFreactor()

# simple settings
pdfReactor.setAddBookmarks(True)
pdfReactor.setAddLinks(True)

# adding a user style sheet
pdfReactor.addUserStyleSheet("p { color: red }", "", "", "")

# create PDF and specify the document
result = pdfReactor.renderDocumentFromURL("http://www.realobjects.com");
```

New API

```
pdfReactor = PDFreactor()
config = {
    # specify the document
    'document': "http://www.realobjects.com",

    # simple settings
    'addBookmarks': True,
    'addLinks': True,

    # adding a user style sheet
    'userStyleSheets': [
        {
            'content': "p { color: red }"
        }
    ]
}

# create PDF
# ...as base64 encoded String
result = pdfReactor.convert(config)
pdf = result['document']

# ...as binary
pdf = pdfReactor.convertAsBinary(config)
```

Note: To convert the base64 encoded document into binary data, you can do the following:

```
import base64
print(base64.b64decode(result['document']))
```

PDFReactor Migration Guide

Ruby

Old API

```
pdfReactor = PDFReactor.new();

# simple settings
pdfReactor.setAddBookmarks(true)
pdfReactor.setAddLinks(true)

# adding a user style sheet
pdfReactor.addUserStyleSheet("p { color: red }", "", "", "")

# create PDF and specify the document
result = pdfReactor.renderDocumentFromURL("http://www.realobjects.com")
```

New API

```
pdfReactor = PDFReactor.new()
config = {
  # specify the document
  document: "http://www.realobjects.com",

  # simple settings
  addBookmarks: true,
  addLinks: true,

  # adding a user style sheet
  userStyleSheets: [
    {
      content: "p { color: red }"
    }
  ]
}

# create PDF
# ...as base64 encoded String
result = pdfReactor.convert(config)
pdf = result["document"]

# ...as binary
pdf = pdfReactor.convertAsBinary(config)
```

Note: To convert the base64 encoded document into binary data, you can do the following:

```
require "base64"
print Base64.decode64(result["document"])
```


PDFreactor Migration Guide

Perl

Old API

```
my $pdfReactor = PDFreactor -> new();

# simple settings
$pdfReactor -> setAddBookmarks('true');
$pdfReactor -> setAddLinks('true');

# adding a user style sheet
$pdfReactor -> addUserStyleSheet("p { color: red }", "", "", "");

# create PDF and specify the document
$result = pdfReactor -> renderDocumentFromURL("http://www.realobjects.com");
```

New API

```
my $pdfReactor = PDFreactor -> new();
$config = {
    # specify the document
    'document' => "http://www.realobjects.com",

    # simple settings
    'addBookmarks' => 'true',
    'addLinks' => 'true',

    # adding a user style sheet
    'userStyleSheets' => [
        {
            'content' => "p { color: red }"
        }
    ]
};

# create PDF
# ...as base64 encoded String
$result = $pdfReactor -> convert($config);
$pdf = $result->{'document'};

# ...as binary
$pdf = $pdfReactor -> convertAsBinary($config);
```

Note: To convert the base64 encoded document into binary data, you can do the following:

```
use MIME::Base64 ();
print MIME::Base64::decode($result->{'document'});
```