Documentation

Interface PDF-AS – External Web-Application

Extension of PDF-AS by an Interface to Provide Connectivity to External Web-Applications

Version 1.1, 29.01.2009

Thomas Zefferer – thomas.zefferer@iaik.tugraz.at

Abstract:

This project aimed at extending PDF-AS by an interface that can be used by external web-applications in order to benefit from the functionality provided by PDF-AS. This way, the integration of PDF signing tasks in external web-applications is attempted to be simplified. The implemented interface supports the transmission of arbitrary PDF documents to PDF-AS, where those documents are signed and finally returned to the calling external web-application for further processing.

Table of Contents:

Revision History

Version	Date	Author(s)	
0.1	24.07.2007	Thomas Zefferer	German version
0.2	21.01.2008	Thomas Zefferer	Translation
1.0	21.01.2008	Thomas Zefferer	Final Adaptations
1.1	29.01.2009	Thomas Knall	Documentation of some enhancements added.

1 Initial Situation and Objective Target

PDF-AS is an application that can be used to sign arbitrary PDF documents electronically as well as to verify existing signatures of already signed PDF documents. The functionality of PDF-AS can be accessed via the command line as well as via a web-application that provides an interface to upload the PDF documents to be processed. PDF-AS does also provide an API that can be used to access the application's basic functionality. The resulting signature itself can be adapted by several parameters. For instance, these parameters can be used to switch between the application of textual and binary signatures, or to accommodate the appearance or the position of the signature block that is to be created.
The goal of this project was the specification and implementation of an interface, by which PDF-AS can be accessed easily by external web-applications in order to enable these applications to transmit arbitrary PDF documents to PDF-AS for further processing.

Since this interface has been developed during an ongoing project, the current implementation of this interface is strongly custom-tailored to the special requirements of this project. Especially the set of parameters that are transmitted to PDF-AS depends heavily on the needs of the mentioned project. Anyways, an adaption of the currently available version should be feasible without much effort.

2 Specification of the Interface

Figure 1 illustrates the basic structure of the implemented interface.

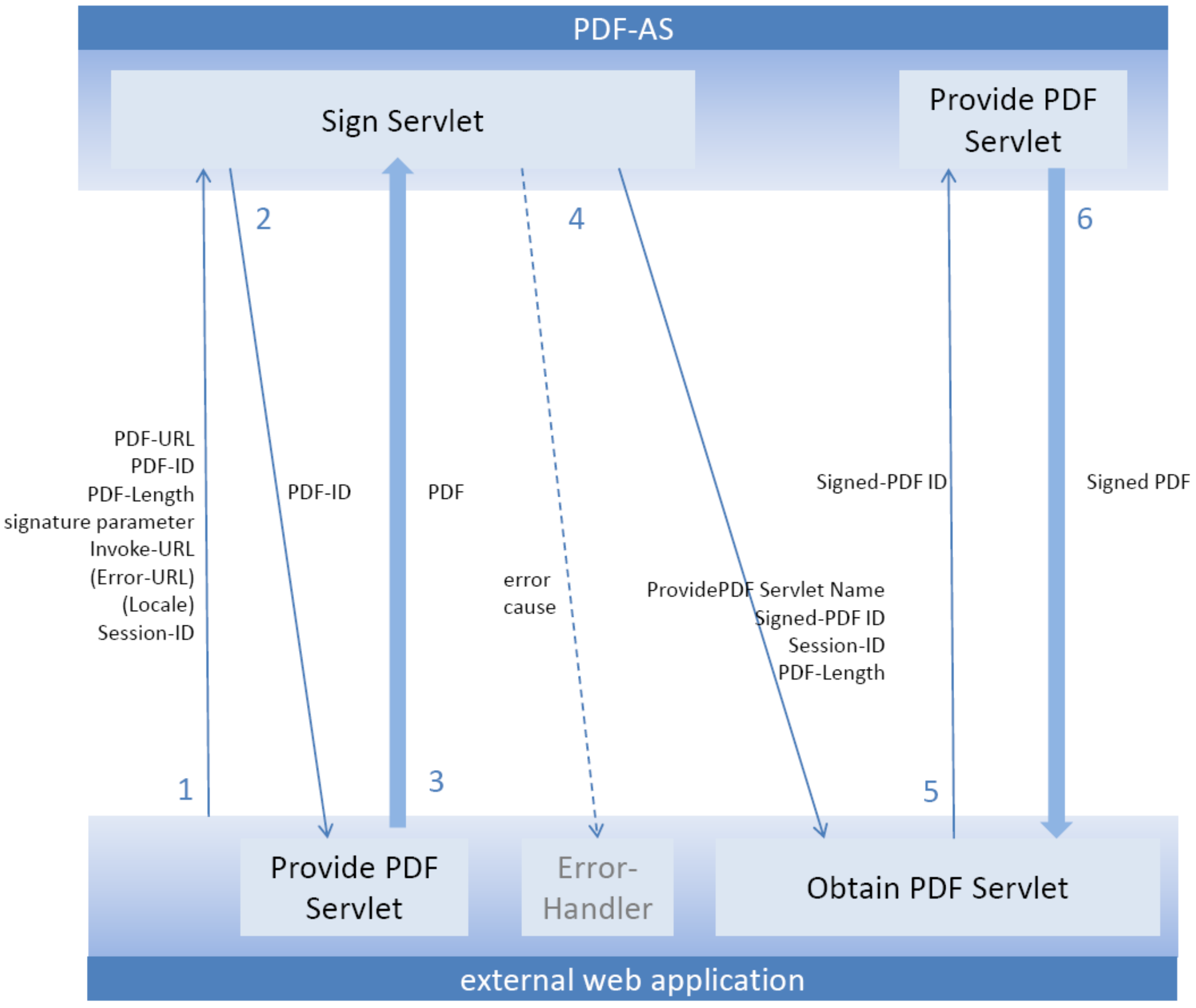

Figure 1 - Interface between PDF-AS and an external web-application

The different steps of the communication between PDF-AS and an external web-application are numbered serially from 1 to 6 and represent the sequence of data interchange between the participating components. The six steps of the communication process using the specified interface are described below.

1. The external web-application, which attempts to benefit from the functionality of PDF-AS in order to sign a PDF document, redirects to the Sign-Servlet of PDF-AS. At that time, the calling web-application has to ensure, that it provides the PDF file to be signed under an URL. This URL is transmitted to PDF-AS as a redirect parameter. There are several other parameters defined as well. They are for instance used to transmit the ID of the document to be signed or several other signature parameters that are required by PDF-AS in order to create the desired electronic signature. Since the calling web-application has to be invoked after the completion of the signing process, the session-ID of the external web-application and the URL, under which the external web-application can be continued are transmitted to PDF-AS too. Apart from

that the optional parameter "locale" may be used to set the language for PDF-AS ("`locale=en`" or "`locale=de_AT`" for instance). Another optional parameter is "`invoke-app-error-url`" which can be used to declare the error handler url of the calling application.

2. PDF-AS fetches the document to be signed using the URL and the ID that have been provided in the first communication step.

3. The respective document to be signed is transmitted to PDF-AS.

4. PDF-AS signs the document. The user is redirected to the external web-application, where the original session is continued using the session-id that has been transmitted to PDF-AS in the first step. PDF-AS does also transmit the ID of the signed document and the name of the Servlet that has to be called by the external web-application in order to obtain the signed PDF file from PDF-AS. The entire URL under which the signed document is provided by PDF-AS does also depend on the environment, in which PDF-AS is running. Hence, only the name of the Servlet, that is responsible for providing the respective data is returned by PDF-AS, while the rest of the URL has to be completed by the calling external web-application itself. The ID of the provided signed PDF document corresponds to the ID of the document that has been transmitted to PDF-AS during steps two and three. If an error handler URL has been declared upon invocation of PDF-AS the user is directed to the error handler URL in case of an error. Thereby the error is url-encoded within the parameter "error" and the cause is encoded within the parameter "cause".

5. The external web-application requests the signed PDF document using the URL and ID that has been provided in step 4.

6. The signed PDF document is finally transmitted to the external web-application.

Note: The current implementation provides the signing of PDF documents only. Nevertheless, the basic principle of this interface could also be used to transmit already signed PDF documents in order to verify present signatures.

3 Implementation

In the following, the implementation of the interface, which has been specified in Section 2 is described. As already mentioned, this implementation is custom-tailored to the requirements of a specific project. Nevertheless, this implementation clarifies the functionality and the usage of the specified interface.

3.1 Extension of PDF-AS

Before the interface being specified in Section 2 had been implemented, there were basically two ways to interact with PDF-AS.

1. Using parameters if calling PDF-AS from the command line
2. Using a web form if PDF-AS is operated as a web-application

3.1.1 Class SignServlet

This class has originally been called by the web form, which is used for the communication between user and PDF-AS, exclusively. In this class, all user-defined parameters have been collected, prepared, and forwarded to the classes that perform the signing process.
Since this class should also be callable by an external web-application, the existing servlet has been extended. When being called, the servlet checks whether the URL of an external web-application is present as a parameter. If this is the case, all further required parameters are obtained from the redirect instead of the web form.
This class is also responsible for the further processing of the signed document. Thus, this part of the servlet has been adapted as well. If the URL of an external web-application is present, the servlet invokes the external web-application using the provided URL. Finally, the signed PDF document is transmitted back to the external web-application.

The following table summarizes the parameters that are necessary for a successful call of the respective Servlet. Some parameters are irrelevant for the signing process. Nevertheless, they have been included to the specification of the interface in order to simplify further extensions and to minimize necessary adaptations of already existing classes of PDF-AS.

Table 1 - Required parameters

Parameter name	Values	Description / Comments
connector	<moa|bku>	For this implementation, only the value "bku" has been tested yet.
filename	<FILENAME>	For the signing of documents irrelevant too, but can be useful in debug scenarios.
inline	<true|false>	For the signing of documents irrelevant. Is provided for the sake of completeness and should be set to "false".
invoke-app-url	<URL>	Defines the URL, where PDF-AS redirects to after completion of the signing process in order to continue the external web-application.

Parameter name	Values	Description / Comments
invoke-app-error-url	<URL>	OPTIONAL: Defines the URL, where PDF-AS redirects the user to in case of an error, providing the error message within the URL-encoded parameter "error" and the cause within the URL-encoded parameter "cause".
locale	<Locale>	OPTIONAL: Defines the language PDF-AS is intended to use. The locale has to be declared using a two-letter language code as defined by ISO-639 together with an optional ISO-3166 country code. (Examples: "de_AT", "de", "en_US", "en")
mode	<textual|binary>	Defines the type of signature.
num-bytes	<Integer>	Defines the number of bytes to be transmitted, i.e. the size of the document.
pdf-id	<Long>	Unique ID, which identifies the document to be signed.
pdf-url	<URL>	Defines the URL where PDF-AS can obtain the document to be signed.
preview	false	This parameter has to be set to false in order to guarantee a correct interchange of data.
session-id	<SESSION-ID>	The session-id of the external web-application. Is required to continue the calling web-application after completion of the signing process.
sig_type	<PROFILENAME>	Defines the name of the PDF-AS profile to be used. The profile defines the appearance of the resulting signature block.
sig-pos-p	<Integer>	Defines the number of the page, on which the signature block is to be created.
sig-pos-y	<Integer>	Defines the vertical position of the signature block to be created.

The following example shows the usage of the parameters described above:

```
https://server.foo/pdf-as/Sign
   ?preview=false
   &connector=bku
   &mode=textual
   &sig_type=SIGNATURBLOCK_DE
   &inline=false
   &filename=test.pdf
   &num-bytes=45916
   &pdf-url=http://localhost:8080/myapp/ProvidePDF
   &pdf-id=1956507909008215134
   &invoke-app-url=https://server.foo/myapp/ReturnSignedPDF
   &invoke-app-error-url=https://server.foo/myapp/ErrorHandler
   &session-id=9085B85B364BEC31E7D38047FE54577D
   &locale=de
```

Note that all parameters have to be provided in URL-encoded manner:

```
https://server.foo/pdf-as/Sign
  ?preview=false
  &connector=bku
  &mode=textual
  &sig_type=SIGNATURBLOCK_DE
  &inline=false
  &filename=test.pdf
  &num-bytes=45916
  &pdf-url=http%3A%2F%2Flocalhost%3A8080%2Fmyapp%2FProvidePDF
  &pdf-id=1956507909008215134
  &invoke-app-url=https%3A%2F%2Fserver.foo%2Fmyapp%2FReturnSignedPDF
  &invoke-app-error-url=https%3A%2F%2Fserver.foo%2Fmyapp%2FErrorHandler
  &session-id=9085B85B364BEC31E7D38047FE54577D
  &locale=de
```

3.1.2 Class SessionInformation

This class encapsulates all information that is provided by the user when PDF-AS is operated as a web-application. When PDF-AS is called by an external web-application, even more data is provided to PDF-AS. Hence, the existing class has been extended by two entries.

1. A TablePos object stores information about the desired position of the signature block that is to be created. In the current version the vertical position and page, on which the signature block should be placed, can be defined.
2. In an ExternAppInformation object all available information about the calling external web-application is stored. A more detailed description of this class is given in Section 3.1.3

3.1.3 Class ExternAppInformation

This class has been added to the application and encapsulates information about the calling external web-application. This class stores the session-id and the URL, which are both used to continue the external web-application after completion of the signing process, as well as the unique ID of the document to be signed.

3.1.4 Class ProvidePDFServlet

The class ProvidePDFServlet has been added to the existing PDF-AS source as well. When this Servlet is called, the document with the given ID is searched in all already signed and stored PDF documents. When the document has been transmitted to the external web-application, the respective document is deleted.

In the following table, the parameter that is required for a successful call of this Servlet is described.

Table 2 - Required parameter

Parameter name	Value	Description / Comment
pdf-id	<Long>	Unique ID, which identifies the document to be transmitted.

3.1.5 Class PDFContainer

The class PDFContainer contains a signed PDF document and its respective ID. This class is used in order to assign unique IDs to already signed documents and to store them until they are fetched from an external web-application.

3.2 Connecting External Web-Applications to PDF-AS

In order to guarantee a correct communication with PDF-AS, the external web-application has to provide two Servlets that can be called by PDF-AS. As the URLs to these Servlets are transmitted to PDF-AS as parameters anyways, the names of the Servlets that are given here are not obligatory at all.

3.2.1 ProvidePDFServlet

Comparable to the PDF-AS Servlet of the same name, this class provides a method that can be used to obtain a PDF document with a specific ID. In order to ensure a successful data transfer, the URL to this Servlet and the ID of the document to be signed have to be provided to PDF-AS as parameters. The file itself is then streamed using the output stream of the Servlet's response parameter in order to transmit it to PDF-AS.

The following table describes the parameter, which is required for a successful call of the Servlet.

Table 3 - Required parameter

Parameter name	Value	Description / Comment
pdf-id	<Long>	Unique id that identifies the PDF document to be transmitted.

3.2.2 ObtainPDFServlet

Using this Servlet, the external web-application is continued by PDF-AS using the session-id that has been transmitted during the first step of the communication. When this Servlet is called by PDF-AS, the name of the Servlet that has to be called in order to obtain the signed document is transmitted as well. With this name and the knowledge about the location of the used PDF-AS instance, the entire URL, under which the signed document can be obtained, has to be reconstructed first. Using this URL and the provided ID of the document to be fetched, the signed PDF document can finally be obtained. The respective PDF-AS Servlet provides the file data using the output stream of its response parameter. Hence, the signed PDF document can be read in over an input stream from the given URL.

The following table summarizes all parameters, which are required for a successful call of this Servlet.

Table 4 - Required parameters

Parameter name	Value	Description / Comments
num-bytes	<Integer>	Defines the number of bytes to be transmitted, i.e. the size of the document.
pdf-id	<Long>	Unique ID that identifies the document to be transmitted.
pdf-url	<URL>	Defines the name of the PDF-AS Servlet, which can be called to obtain the signed document.
session-id	<SESSION-ID>	Used to continue the original session.