Nets DanID A/S
Lautrupbjerg 10
DK – 2750 Ballerup

T +45 87 42 45 00
F +45 70 20 66 29
www.nets.dk

CVR no. 30808460

Specification document for PDF Validator API

Table of Contents

Version history

21 August 2009	Version 1	MMART
12 December 2016	Version 1.1	KMAIB

1. Purpose and target group

This document is a part of the NemID Service Provider Package.

The purpose of the document is to provide a general introduction to the PDF Validator API and how to use the API to validate PDF files programmatically.

The document is aimed at the people at the service provider who are responsible for the implementation of NemID.

2. Introduction

When a PDF document is to be signed using either the NemID CodeFile client, it will first be validated. For security reasons not all features of the PDF specification are supported when signing PDF documents.

Using the PDF Validator API, it is possible to validate PDF documents programmatically, to ensure they can be signed using the NemID CodeFile client.

3. API

This section describes the API and how to use it.

1.1. Required files

To use the PDF Validator API from another Java application, you need make sure the following jar files is added to the class path:

- lib/pdf-validator-engine-x.y.z-SNAPSHOT.jar

x.y.z specifies the current version of the OpenSign distribution.

1.2. Usage

There are two ways to run the validation:

1. With *fast-fail* set. This means that the validation will stop once the first error is encountered.
2. Without *fast-fail set*. This means that the entire PDF file will be validated and errors encountered will be collected, and may be inspected later using the ErrorHandler class.

The interface PDFValidator is the primary interface when validating PDF files. To access an implementation of this interface, you use the PDFValidatorFactory class. This class supports 2 different ways of accessing a PDFValidator implementation. The following example simply returns the latest version of the validator:

```
PDFValidator validator =
  PDFValidatorFactory.getInstance().getPDFValidator();
```

Notice that the PDFValidator API supports multiple versions. This means that when a new PDFValidator is, it includes prior versions of the PDF Validator. That way it is possible to validate PDF files using different versions of the PDF Validator. The following example shows how to access a specific version of the PDF Validator:

```
PDFValidator validator =
  PDFValidatorFactory.getInstance().getPDFValidator("2.0.0");
```

To get a list of all supported PDF Validator versions, use:

```
List<String> supportedVersions =
PDFValidatorFactory.getAllVersions();
```

1.3. Example using fast-fail

The following example shows how to validate using fast-fail. First a PDF Validator instance is created by specifying a validator version. Then the validation is performed and eventually errors (the first) can be handled:

```
// First access the validator implementation
PDFValidator validator =
  PDFValidatorFactory.getInstance().getPDFValidator();

// Now point out a file to validate
File fileToValidate = new File("my-pdf-file.pdf");

// Validate the file using fast-fail option
try {
            validator.readPDF(fileToValidate, true);
} catch (Exception e) {
  // Validation failed - handle error
}
```

1.4. Example without fast-fail

The following example shows how to validate without fast-fail. First a PDF Validator instance is created by specifying a validator version. Then the validation is performed and eventually errors (all of them) can be handled using the ErrorHandler class:

```
// First access the validator implementation
PDFValidator validator =
  PDFValidatorFactory.getInstance().getPDFValidator();

// Now point out a file to validate
File fileToValidate = new File("my-pdf-file.pdf");

// Validate the file using fast-fail option
try {
  validator.readPDF(fileToValidate, false);
} catch (Exception e) {
  // Validation failed - handle error(s)

  // Get the error handler
  ErrorHandler errorHandler = validator.getErrorHandler();

  // Get the list of errors (PDFEXception objects)
  List<PDFException> errors = errorHandler.getErrorsFound();
}
```