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-nr. 30808460

Specifikationsdokument for PDF Validator API

Indholdsfortegnelse

1 Formål og målgruppe 4

2 Introduktion 5

3 API 6
3.1 Nødvendige filer 6
3.2 Anvendelse 6
3.3 Eksempel med fast-fail 7
3.4 Eksempel uden fast-fail 8

Versionsfortegnelse

17. august 2012	Version 1.0	MMART
12. december 2016	Version 1.1	KMAIB

1 Formål og målgruppe

Dette dokument er en del af NemID tjenesteudbyderpakken.

Dokumentet vil give beskrive brugen af PDF Validator API'et. Med dette API er det muligt at foretage programmatisk validering af PDF dokumenter.

Dokumentet henvender sig til de personer hos tjeneste-udbyderen, der er ansvarlige for implementeringen af NemID.

2 Introduktion

Når et PDF dokument skal signeres via NemID CodeFile-klienten, foretages der først en validering af dokumentet. Det er ikke alle funktioner i PDF specifikationen, som – af sikkerhedsmæssige årsager – er tilladt i PDF dokumenter, der skal signeres.

Med PDF Validator API er det muligt at validere PDF dokumenter programmatisk for at sikre, at de vil kunne signeres med NemID CodeFile-klienten.

3 API

Følgende beskriver PDF Validator API'et, og hvordan det anvendes.

3.1 Nødvendige filer

For at bruge valideringsapplikationen fra en anden Java applikation, skal følge jar-fil tilføjes til classpath:

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

x.y.z angiver den aktuelle version af OpenSign distributionen.

3.2 Anvendelse

Der er 2 måder at kalde valideringen på:

1. Med *fast-fail* angivet. Dette betyder, at valideringen stopper når den første fejl opdages.
2. Uden *fast-fail*. Dette betyder, at hele PDF filen valideres og fejl opsamles undervejs og kan efterfølgende undersøges via en ErrorHandler klasse.

Interfacet PDFValidator er den primære grænseflade, der skal bruges for at valideres PDF-filer. En implementation af klassen fås via klassen PDFValidatorFactory. Denne klasser understøtter 2 forskellige måder at få en PDFValidator. Nedenstående returnerer den nyeste version af PDFValidator:

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

Bemærk at PDFValidator applikationen understøtter flere versioner. Det betyder, at når en ny PDFValidator frigives, så indeholder den tidligere versioner af PDFValidator. Dermed er det muligt at validere PDF-filer mod forskellige versioner af PDF-Validator. Følgende eksempel viser hvordan en bestemt version af PDFValidator fås:

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

For at se hvilke versioner, der er understøttet kaldes:

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

3.3 Eksempel med fast-fail

Følgende eksempel viser en validering med fast-fail. Først konstrueres en PDFValidator ved at angive hvilken version, der ønskes. Herefter kaldes selve valideringen og eventuelle (første) fejl kan håndteres:

```
// 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
}
```

3.4 Eksempel uden fast-fail

Følgende eksempel viser en validering uden fast-fail. Først konstrueres en PDFValidator ved at angive hvilken version, der ønskes. Herefter kaldes selve valideringen og eventuelle fejl (alle) kan håndteres via error handleren:

```
// 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();
}
```