Nets DanID A/S
Lautrupbjerg 10
DK – 2750 Ballerup

T +45 87 42 45 00
F+45 70 20 66 29
info@danid.dk
www.nets-danid.dk

CVR-nr. 30808460

Implementerings-vejledning for NemID

Indholdsfortegnelse

Versionsfortegnelse

25. maj 2009	Version 1.0	MOBO
10. juli 2009	Version 1.1	MOBO
24.februar 2010	Version 1.2	MOBO
10. juni 2010	Version 1.3	MTV
17. september 2010	Version 1.4	UFS + HENR
29. oktober 2010	Version 1.5	MTV
9. december 2010	Version 1.6	MTV
5. januar 2011	Version 1.7	MTV
20. januar 2011	Version 1.8	MTV
9. marts 2011	Version 1.9	JV
12. april 2011	Version 2.0	SHP
15. juni 2011	Version 2.1	JV
25. oktober 2011	Version 2.2	MTVOL
9. oktober 2012	Version 2.3	BMATZ
15. november	Version 2.4	BMATZ

1 Dokumentets formål og målgruppe

Dokumentets formål er, at hjælpe tjenesteudbydere med at integrere Nets DanIDs funktionalitet til autentificering ind i egne systemer i forbindelse med implementeringen af NemID.

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

Oversigt over alle dokumenter i Tjenesteudbyderpakken:

Overordnet dokumentation

- Introduktion til NemID og Tjenesteudbyderpakken
- Anbefalinger til interaktionsdesign og brugervalg af applet
- Drejebog for migrering til NemID
- Termer og begreber i NemID

Implementeringsdokumentation

- **Implementeringsvejledning for NemID**
- Konfiguration og opsætning

Testdokumentation

- Vejledning i brug af test tools
- Anbefalede testprocedurer

Referencedokumentation

- Specifikationsdokument for servicen PID-CPR
- Specifikationsdokument for servicen RID-CPR
- Specifikationsdokument for LDAP API
- Specifikationsdokument for OCSP
- Specifikationsdokument for OCES II

2 Dokumentets opbygning

Dokumentet beskriver, hvordan der integreres til de to NemID-appletter: **Applet med OTP** og **Applet uden OTP** og efterfølgende får valideret svaret fra appletterne.

Dokumentets opbygning:

- Afsnit 3 beskriver integrationen til Applet med OTP.
- Afsnit 4 beskriver signering af PDF dokumenter og øvrige bilag
- Afsnit 5 beskriver integrationen til Applet uden OTP.
- Afsnit 6 beskriver, hvordan svaret fra appletterne til webserveren skal valideres. Gennemgangen omfatter såvel Applet med OTP og Applet uden OTP.
- Afsnit 8 indeholder referenceoplysninger for tjenesteudbydere, der ønsker direkte integration til NemID.

Ved valg af integrationsmetode til NemID anbefales det, at tjenesteudbydere tager udgangspunkt i Nets DanIDs OOAPI. Dermed dækkes de fleste integrationsbehov.

Såfremt en tjenesteudbyder ønsker at videreføre sin nuværende specialtilpassede løsning eller benytter sig af funktioner, som OOAPI'et ikke dækker, henvises til dokumenterne i mappen **Referencedokumentation**, der indeholder oplysninger om direkte integration til infrastrukturen.

3 Integration til Applet med OTP

3.1 Standarder og algoritmer

Dette afsnit præsenterer de algoritmer og standarder der benyttes i NemID-systemet.

3.1.1 XMLDSig

`XMLDSig` (XML Signature Syntax and Processing) er en W3C-standard, som benyttes til at transmittere information om digitale underskrifter.

Standarden er siden dens lancering blevet udvidet med dokumentet `RFC 4051`. Den vigtigste tilføjelse i RFC 4051 er muligheden for at benytte algoritmerne `SHA-256` og `SHA-512` ved signering.

3.1.2 Kryptografiske algoritmer

Med mindre andet er angivet under beskrivelsen af specifikke funktioner, benyttes nedenstående kryptografiske algoritmer:

Hash	SHA-256
Signering	RSA with 2048 bit keys

3.2 Overblik over kommunikationen

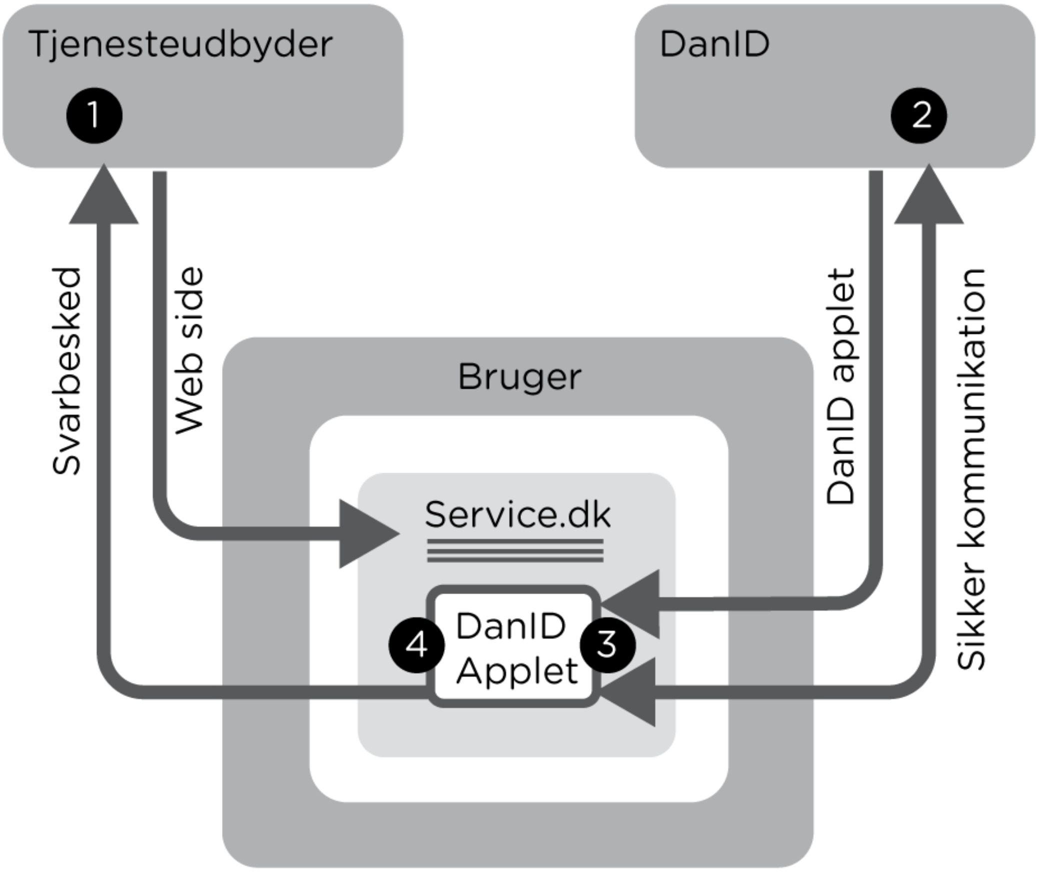
Kommunikation ved log-in og signering er som følger:

1. Tjenesteudbyderen genererer en webside og overfører den til brugeren. Siden refererer til en applet, som er placeret hos Nets DanID.

2. Nets DanID overfører appletten til brugeren. Appletten læser de parametre, tjenesteudbyderen har sat, og afgør ud fra dem, hvilken funktion der skal udføres.

3. Appletten kommunikerer over en sikker forbindelse med Nets DanID for at fastslå brugerens identitet.

4. Resultatet af log-in eller signeringen sendes fra appletten tilbage til tjenesteudbyderen.



Figur 1 - Diagram over kommunikationen ved log-in og signering.

3.3 Opsætning af Applet med OTP

For at starte denne applet, skal tjenesteudbyderen generere en webside med et applet-tag indeholdende en normaliseret parameterliste.

Efter normaliseringen beregnes den normaliserede strengs SHA-256 hash-værdi, og resultatet heraf signeres med tjenesteudbyderens VOCES-certifikat. Både strengens hash-værdi og signatur sendes til appletten som parametre.

Såfremt tjenesteudbyderen ikke har et VOCES-certifikat, vil Nets DanID gratis udstede et til tjenesteudbyderen, når der er indgået en Tjenesteudbyderaftale med Nets DanID.

Tjenesteudbyderpakken indeholder bl.a. følgende elementer der kan hjælpe tjenesteudbydere med at få løsningen sat op:

1. .Net og Java-referencekode til generering af applet-tags og signering af parametrene.
2. Eksempler på hvordan denne .Net og Java-kode kan inkluderes i en tjenesteudbyders webside.
3. .Net-doc og Java-doc til beskrivelse af koden.

3.3.1 Inkludering af applet

Appletten inkluderes på siden via HTML-elementet `<applet>`. Appletten skal være 200 pixels bred og 250 pixels høj. Dens startklasse hedder `dk.nemid.OcesApplet`.

Inkludering af appletten i siden kan gøres som i følgende eksempel, eller ved brug af den inkluderede klasse `OcesAppletElementGenerator` (se afsnit 3.6 for yderligere oplysninger om denne klasse).

```
<applet name="DANID_DIGITAL_SIGNATUR" tabindex="1" archive="https://systemgenerated url"
code="dk.pbs.applet.bootstrap.BootApplet" WIDTH="200" HEIGHT="250" mayscript="mayscript">
```

`codebase`-attributten skal indeholde stien til appletten relativt til den side, hvori elementet inkluderes. `mayscript` tillader appletten at kommunikere med browseren, så den kan sende svar til tjenesteudbyderen.

For at appletten kan indsende NemIDs svar til tjenesteudbyderen, skal siden også indeholde et`<form>`-element.

```
<form name="signedForm" method="post" action="">
<input type="hidden" name="signature">
<input type="hidden" name="result">
</form>
```

For at appletten kan indsætte returværdierne i ovenstående form, skal disse javascript-funktioner også inkluderes i siden.

```
<script language="JavaScript">
  function onSignOK(signature) {
    document.signedForm.action = "response.jsp";
    document.signedForm.signature.value=signature;
    document.signedForm.result.value='ok';
    document.signedForm.submit();
  }
  function onSignCancel() {
    document.signedForm.action = "response.jsp";
    document.signedForm.result.value='cancel';
    document.signedForm.submit();
  }
  function onSignError(msg) {
    document.signedForm.action = "response.jsp";
    document.signedForm.result.value=msg;
document.signedForm.submit();
}
</script>
```

3.4 Størrelse af applet

Standard NemID applet er 200 pixels bred og 250 pixels høj.

Hvis større skærmbillede er nødvendigt, for eksempel for at gøre det muligt for brugeren at ændre sin adgangskode, anvendes en pop-up dialog til at vise de større skærmbilleder.

Hvis tjenesteudbyderen fastslår, at pop-up dialogbokse er uønskede, kan appletten bruges i always_embedded tilstand. Dette vil deaktivere pop-up-funktionalitet, og køre alt inden for applettens område. Når du bruger denne mulighed, skal den anbefalede størrelse for appletten være 500 pixels bred og 450 pixels høj. Dette kan justeres ved hjælp af signWidth og signHeightparameterne, hvis det er nødvendigt.

Parameteren background_color kan bruges til at styre farven på det område, der i øjeblikket ikke bruges til at tegne applet komponenter.

Som standard vises signerings skærmene i en pop-up dialogboks, der er 600 pixels bred og 350 pixels høj. HTML-elementet der dikterer størrelsen af appletten, skal have bredde og højde sat til 0, da alle skærmene er vist i pop-up dialogbokse i løbet af signeringen.

Pop-up dialogbokse vises som standard uden vinduesdekorationer såsom close-knapper og titel-baren. Dekorationerne kan aktiveres ved hjælp af boolean WindowDecorated parameter.

3.5 Parametre

I dette afsnit beskrives de parametre, som bruges til at styre applettens opførsel. Applet parameter navne er case insensitive. Alle parametre, som ikke udtrykkeligt er beskrevet som obligatoriske er valgfrie.

I dette afsnit menes med "parametre" de værdier, der kan angives i de `<param>` tags, som kan angives som sub-elementer til et `<applet>` tag. Der menes *ikke* attributterne på `<applet>` tagget.

Parametre skal være signerede for at sikre applet integriteten. Se afsnit 3.5.3 for yderligere detaljer.

3.5.1 Fælles parametre

Dette afsnit beskriver de parametre, der bruges med applet, uanset om der skal foretages autentifikation eller signering.

Zip_File_Alias **obligatorisk**
Denne parameter bestemmer operation, som applet vil udføre. Parameteren er obligatorisk og kan antage følgende værdier

`OpenLogon2`	Initiates an OCES authentication flow.
`OpenSign2`	Initiates an OCES signing flow.

ServerUrlPrefix **obligatorisk**
Angiver adressen på Nets DanID server, som appletten skal kommunikere med. Denne parameter er obligatorisk. Det vil sandsynligvis blive fjernet i produktion, og dens værdi hard-kodet i appletten.

MayScript **obligatorisk**
Denne parameter skal medtages for at sikre, at appletten kan kommunikere med browseren og vende tilbage med svar til tjenesteudbyderen. Det er obligatorisk og skal altid have værdien "true".

Bemærk: denne parameter SKAL udelukkes fra generation af parameter integritet værdier på grund af browser uoverensstemmelser.

ParamCert **obligatorisk**
Tjenesteudbyderens OCES-certifikat skal medsendes i denne parameter. Certifikatet skal bruges til at validere signaturen på applettens parametre, og til at inkludere tjenesteudbyderens navn i appletten.

ParamsDigest **obligatorisk**
Digest af parametre til appletten. Denne parameter er obligatorisk og er yderligere beskrevet i afsnit3.5.3.

SignedDigest **obligatorisk**
Underskriften på parametre til appletten. Denne parameter er obligatorisk og er yderligere beskrevet i afsnit 3.5.3.

Log_Level
Sætter logniveau af beskeder der skrives ud javakonsollen. Gyldige værdier er INFO, DEBUG og FEJL. Standard niveau er FEJL.

Background_color
Hvis det areal, der er afsat til appletten er større end den synlige størrelse vil farven i det tilbageværende rum være hvad der er angivet i denne parameter. Parameteren skal være en hexadecimal RGB værdi af formatet # RRGGBB.

always_embedded
Denne parameter bestemmer, om der skal bruges popup-dialoger til at vise nogle af de skærme, der kan vises under godkendelse (f.eks skærme for aktivering eller indstilling af password). Parameteren er valgfri og kan antage værdierne "true" eller "false".
Betydningen af den parameter er nærmere uddybet i afsnit 3.4.

WindowDecorated
Hvis man vil tilføje vinduesdekorationer (fx en titel bar) til en pop-up skal denne parameter sættes til true.

language
Konfigurerer hvilket sprog til brug i applet. Applet standard er Danish, hvis parameteren er udeladt.

De tilladte parameter værdier

`EN`	Engelsk
`KL`	Grønlandsk
`DA`	Dansk (standard valg, hvis parameteren er udeladt)

3.5.2 Signing parametre

I dette afsnit nævnes de parametre, der styrer signeringsfunktionaliteten i appletten. Afhængig af hvilket format signeringsteksten er i (klartekst, HTML,XML, eller PDF) kan der være obligatoriske parametre der er specifikke for formatet.

SignText **obligatorisk (ikke for PDF)**
Den base64 kodet tekst, der skal underskrives af brugeren. Teksten skal være i et af de formater, der er skitseret i afsnit 4.

SignTextFormat **obligatorisk**
Angiver mime type signText parameter. Mulige værdier er PLAIN, HTML, XML eller PDF.

For PDF dokumenter skal der yderligere angives oplysninger til applet hvor den skal hente det pågældende dokument. Dette gøres i form af følgende parametre:

- signtext.uri – Angiver URI til PDF dokument. PDF dokument skal ligge på samme server som HTML applet tag er indlejret i. URI'en angives som en relativ URL i forhold til hosten.

- signtext.hash.value - En BASE64 enkodet hash værdi af PDF dokumentet. For OTP-applet benyttes sha-256 hash algoritmen.

- signtext.hash.algorithm – For OpenSign applet skal der angives hvilken hash algoritme der anvendt til at beregne hash værdi af PDF dokument. OpenSign supporterer sha-1 og sha-256 hash algoritmerne.

SignWidth
Denne valgfri parameter sætter bredden i pixels af popup-dialogboks der viser signeringsteksten for brugeren. Parameteren er uddybet i afsnit 3.4.

SignHeight
Denne valgfri parameter indstiller højde i pixels af popup-dialogboks der viser signeringsteksten for brugeren. Parameteren er uddybet i afsnit 3.4.

SignProperties
Denne parameter indeholder en liste med tjenesteudbyderspecifikke værdier, som skal inkluderes i det underskrevne dokument. Listen indeholder semikolon-adskilte, navngivne værdier.

Et eksempel på dette er:

```
property1=testproperty1;property2=testproperty2
```

Navnene kan kun bestå af bogstaverne fra a til z, underscore og tal. Værdierne bliver base64-indkodede, inden de inkluderes i det signerede dokument og har dermed ingen begrænsninger.

En mulig navngiven værdi er ”challenge” (f.eks. hvor vi angiver en base64-indkodet udgave af teksten ”mychallenge”):

```
    ”challenge=bXljaGFsbGVuZ2U=”.
```

Hvis denne challenge ændres ved hvert login eller signering, kan resultatet fra appletten tjekkes mod denne for at undgå såkaldte ”replayattacks”.

Hvis OOAPI bruges i forbindelse med validering af login eller signering, er denne challenge påkrævet.

3.5.3 Sikring af applet-parametre

Applettens parametre skal signeres af tjenesteudbyderen for at sikre dem imod at blive manipuleret, før de bliver læst af appletten.

Processen til sikring af applettens parametre er:

1. Tjenesteudbyderen samler listen over de parametre, der skal sendes til appletten. Listen normaliseres til en samlet tekststreng (se Afsnit3.5.4 **Normalisering af parametre**). Hash-værdien af denne streng beregnes. Værdien signeres med tjenesteudbyderens nøgle (virksomhedscertifikat - VOCES).

2. Websiden, som indeholder appletten og dens parametre, genereres. Hash-værdien og signaturen fra forrige trin inkluderes som applet-parametre.
3. Appletten læser parametrene og normaliserer dem på samme måde som tjenesteudbyderen. Hash-værdien og signaturen holdes ude fra normaliseringen. Appletten verificerer, at den beregnede hash-værdi er identisk med den værdi, tjenesteudbyderen medsendte som parameter. Hvis dette fejler, afbrydes forløbet, og appletten sender en fejlkode til tjenesteudbyderen.
4. Appletten sender parametrenes hash-værdi og signaturen til Nets DanID over en sikker forbindelse. Nets DanID verificerer signaturen med tjenesteudbyderens certifikat. Hvis signaturen ikke stemmer, afbrydes forløbet, og en fejlkode sendes til tjenesteudbyderen.

Hash-værdien af de normaliserede parametre skal tilføjes som en applet parameter med navnet `‘paramsdigest’`, mens signaturen skal medsendes i en parameter med navnet `’signeddigest’`.

```
<appletcode="org.openoces.opensign.client.applet.Sign" codebase="http://www.danid.dk/sys/"
ARCHIVE="Auth.jar">
<paramname="inputmime" value="text/plain" />
<paramname="SAMLRequest" value="BGEWOGJRWW1241KGFWGFzSxg98yB/MpS6N39UGZFbt9kW [...]" />
<paramname="inputstyle" value="default" />
<paramname="signtext" value="I.O.U.$$$s" />
<paramname="inputstylesheet" value="BASE64ENCODEDXSLTDOCUMENT" />
<paramname="paramsdigest" value="Pehp6s+yFQjyw1wnQyNh2iFvCw0=" />
<paramname="signeddigest"
value="CIw0RDoo7eh1410JUeUJC0ux2ghnDfzSrLDG+V/zSxg98yB/MpS6N39UGZ+XufnxnNihhmEp23MRUkQgRgOFbt9kW
S/wbhBy/F70np/vESMLNjPe3Bp9JcSgDrYqxl4c1WzUnVE9JVehYONOPFXbq9zCorj/vJo9W63DxVW+e/s=" />
</applet>
```

Parametrene i eksemplet ovenfor er kun eksempler, der skal illustrere den beskrevne proces. De er ikke nødvendigvis parametre, der reelt er gyldige ved brug af appletten.

3.5.4 Normalisering af parametre

Hash-værdien af parametrene skal beregnes ud fra en normaliseret form. Den normaliserede form genereres udelukkende for at beregne hash-værdien, så det er ikke nødvendigt for normaliseringsprocessen at være reversibel, dvs. tillade, at parametrene kan læses fra deres normaliserede form.

Fremgangsmåde ved normalisering af applet-parametrene:

1. Parametrene sorteres alfabetisk ud fra deres navn. Sorteringen skelner ikke mellem store og små bogstaver.

2. Parametrene sammenlægges i en streng, som en skiftende sekvens af parameter-navn og parameter-værdi:

   lc(name$_1$) || value$_1$ || lc(name$_2$) || value$_2$ || … || lc(name$_n$) || value$_n$

   Parametrenes navne konverteres til små bogstaver inden sammensætningen.

Den normaliserede form af parametrene i ovenstående eksempel ser således ud:

```
inputmimetext/plaininputstyledefaultinputstylesheetBASE64ENCODEDXSLTDOCUMENTsamlrequestBGEWOGJRW
W1241KGFWGFzSxg98yB/MpS6N39UGZFbt9kWsigntextI.O.U.$$$s
```

Den normaliserede tekst-streng skal være i UTF-8.

3.6 Generering af applet med Sikkerhedspakke

OOAPI'et indeholder .Net og Java-kode, som demonstrerer håndteringen af applet-parametre og svar modtaget fra appletten. I dette afsnit præsenteres klasserne, og deres interfaces dokumenteres kort.

Eksempler på, hvordan komponenterne benyttes kan ses i de jsp-sider, der inkluderes i pakken.

Klassen OcesAppletElementGenerator

Denne klasse genererer `applet`-elementet, som skal indlejres i tjenesteudbyderens hjemmeside.

De vigtigste metoder i klassen er:

public OcesAppletElementGenerator(Signer signer)

Skaber en instans af klassen med en `signer`, der har den private nøgle og kan signere.

public void addSignedParameter(String key, String value)

Tilføjer en parameter til listen, der inkluderes i `applet`-elementet. Parameteren signeres senere i `generateXxxAppletElement`-metoden nedenfor.

public void addUnsignedParameter(String key, String value)

Tilføjer en parameter til listen, der inkluderes i `applet`-elementet. Denne parameter signeres ikke.

public String generateSignAppletElement(String formAction)

Genererer et applet tag til at launche appletten i log-in-mode. Signerer de parametre, der er tilføjet vha. `addSignedParameter`-metoden. Parametrene signeres med den private nøgle i applet-`applet-parameter-signing-keystore-cvr30807460-uid1263281782319.jks`, som der henvises til fra signer-objektet.

public String generateLogonAppletElement(String formAction)

Genererer et applet tag til at launche appletten i signerings-mode. Signerer de parametre, der er tilføjet vha. `addSignedParameter`-metoden. Parametrene signeres med den private nøgle i `applet-applet-parameter-signing-keystore-cvr30807460-uid1263281782319.jks`, som der henvises til fra `signer`-objektet.

Den returnerede `String` indeholder applet HTML-elementet, som vil inkludere appletten i en hjemmeside. Form-elementet vil `post`'e til den `url`, der er angivet i `formAction`-parameteren.

3.6.1 Yderligere referencer

Via følgende links kan du få yderligere nyttige oplysninger:

[XMLDSIG]	"XML Signature Syntax and Processing (Second Edition)"http://www.w3.org/TR/xmldsig-core/
[XMLENC]	"XML Encryption Syntax and Processing" http://www.w3.org/TR/xmlenc-core/
[RFC 4051]	"Additional XML Security Uniform Resource Identifiers (URIs)" http://www.ietf.org/rfc/rfc4051.txt
[PKCS1]	PKCS #1: RSA Cryptography Specifications 2.0 http://tools.ietf.org/html/rfc2437#page-13

3.7 Fejl koder

En fejlkode returneres til den tjenesteudbyder, hvis en applet operation ikke er fuldført. Fejlkoden bør anvendes til at hjælpe brugeren ud af den situation han er kommet i.

Listen indeholder 3 kategorier:

- Generelle fejlkoder
- Fejlkoder i forbindelse med log-in
- Fejlkoder i forbindelse med signeringen.

3.7.1 Generelle fejlkoder

Disse fejlkoder er generelle for applet funktionalitet og kan forekomme i alle situationer, hvor appletten anvendes.

Fejlkode	Årsag til fejl
APP001	Appletten beregnede et digest på dets parametre, og det passer ikke til det digest, der blev fremlagt i parameteren paramsdigest.
SRV001	Underskrift på applettens parametre kunne ikke verificeres af Nets DanID.
SRV004	En uoprettelig fejl opstod hos Nets DanID.
APP003	En uoprettelig intern fejl er opstået i appletten. Stack trace fra denne form for fejl overføres automatisk til Nets DanID til analyse.
CAN001	Brugeren vælger at annullere et aktiveringsflow ved at trykke på knappen "Afbryd". Bemærk at denne fejl ikke indberettes, hvis brugeren navigerer væk fra siden, der indeholder appletten, fx ved at lukke browservinduet eller klikke på et link.
CAN002	Brugeren trykker på knappen "Afbryd" (se CAN001 ved afbrudt aktiveringsflow). Bemærk at denne fejl ikke indberettes, hvis brugeren navigerer væk fra siden, der indeholder appletten, fx ved at lukke browservinduet eller klikke på et link.
LOCK001	Brugeren har indtastet en forkert adgangskode for mange gange, og hans OTP enhed (nøglekort) er nu i karantæne i 8 timer. Denne fejlkode returneres i en session, hvor karantænen er indledt.

LOCK002	Brugeren har indtastet en forkert adgangskode for mange gange, og hans NemID er nu låst. Denne fejlkode returneres i løbet af sessionen, hvor låsen er initieret.
LOCK003	Brugeren har overskredet det tilladte antal indtastning af nøgler ved samme log-in og hans OTP enhed (nøglekort) er nu spærret. Denne fejlkode returneres i løbet af sessionen, hvor spærringen er initieret.
AUTH004	Brugerens NemID er i øjeblikket sat i karantæne, på grund af for mange fejlslagne forsøg. Denne fejlkode returneres, hvis brugeren forsøger at aktiverer en OTP enhed, der er kommet i karantæne i en tidligere session.
AUTH005	Brugerens NemID er låst, på grund af for mange mislykkede forsøg på at indtaste adgangskode. Denne fejl kode kommer, hvis brugeren forsøger at logge på med et NemID, der er blevet låst i en tidligere session.
AUTH006	Brugeren er løbet tør for nøgler på sit nøglekort og har ikke et andet nøglekort der afventer aktivering.
AUTH007	Brugerens NemID adgangskode tilbagekaldes på grund af for mange fejlslagne indtastningsforsøg. Denne fejlkode returneres, hvis brugeren forsøger at godkende et NemID, der har været inddraget under en tidligere session.
AUTH008	Brugerens NemID er ikke aktiveret, og brugeren har ikke en aktiv midlertidig adgangskode.
APP005	Brugeren har valgt ikke at have tillid til certifikatet, der kan verificere signaturen på appletten.
SRV006	Serveren mistede sessionen som var etableret med appletten. Dette kan forekomme, hvis brugeren forlader appletten i et længere

	tidsrum.

3.7.2 Signing fejlkoder

Følgende koder kan returneres, hvis en signeringsoperation ikke lykkedes.

Fejlkode	Årsag til fejl
APP002	Signeringsteksten indeholder ulovlige tegn f.eks fordi HTML-dokumentet indeholdt ulovlige tags. Bruges også under PDF-signering, hvis URI ikke peger på en PDF, hvis hash af PDF ikke er korrekt, eller hvis PDF ikke lever op til whitelisten.
APP006	Kun relevant under PDF-signering : fremkommer hvis dokumentet anvender en af de 14 standard fonte, men fonten ikke er inkluderet i dokumentet og ikke er tilstede i operativ systemet.

3.7.3 OCES fejlkoder

Følgende koder kan være tilbage i løbet af OCES operationer.

Fejlkode	Årsag til fejl
OCES001	Brugeren har fravalgt OCES, men forsøger at logge ind på en tjenesteudbyder, der kræver det.
OCES002	Brugeren har NemID i netbanken og er ikke kvalificeret til at få udstedt et OCES, men forsøger at logge ind på en tjenesteudbyder,

	der kræver det.
OCES003	Den OTP enhed, der bruges til at logge ind, har ikke OCES tilknyttet. Brugeren har en anden OTP enhed med OCES end den der bruges.
OCES004	Brugeren er ikke OCES-kvalificeret som følge af ikke at have et CPR-nummer eller er yngre end 15 år.
OCES005	Returneret i situationer, hvor en ny attest skal udstedes til at fuldføre handlingen, men en teknisk fejl opstod ved at gøre det.

3.7.4 Anbefaling til tekster for bruger-rettede fejlkoder

Fejlkode	**Beskrivelse**	**Tekst i applet**	**Tekstforslag til TU'er + evt. link til DCH**	**Anbefaling til TU**
CAN001	Brugeren trykkede på "afbryd"-knappen i appletten ifb.m. aktivering (altså under brug af midlertidig adgangskode)	Under aktivering advares brugen mod, at han skal benytte ny midlertidig adgangskode næste gang der forsøges logget på.		TU skal sende brugeren et fornuftigt sted hen, givet hvor brugeren er henne i flowet.
CAN002	Brugeren trykkede på "afbryd"-knappen i appletten, alle andre steder end hvor brugeren er i gang med at logge på med midlertidig adgangskode.		Det er ikke muligt for tjenesteudbyderen at give en besked til brugeren.	TU skal sende brugeren et fornuftigt sted hen, givet hvor brugeren er henne i flowet.
LOCK001	Brugeren har indtastet forkert adgangskode for mange gange. Denne fejlbesked fås i den session, hvor brugeren bliver tidslåst.	Brugeren får i applet besked om at hans adgangskode er tidslåst.	Du har angivet forkert bruger-id eller adgangskode 5 gange i træk. NemID er nu spærret i 8 timer, hvorefter du igen vil have 5 forsøg. Du kan logge på igen efter udløbet af den 8 timers spærreperiode. Du kan ophæve spærringen tidligere ved at kontakte NemID-support på tlf. 80 30 70 50	TU'erne får at vide, hvad der står i appletten. Og så kan de vælge selv.
LOCK002	Brugeren har indtastet forkert adgangskode for mange gange. Denne	Brugeren får i applet besked om at adgang er	NemID er spærret og kan ikke bruges. For at få hjælp til dette, kan du kontakte NemID-support på tlf.	

Fejlkode	Beskrivelse	Tekst i applet	Tekstforslag til TU'er + evt. link til DCH	Anbefaling til TU
	fejlbesked fås i den session, hvor brugeren bliver permanent spærret og midlertidig adgangskode er nødvendig.	spærret og han skal have midlertidig adgangskode for at blive låst op.	80 30 70 50	
LOCK003 (tidligere AUT002)	Brugeren har indtastet forkert nøgle for mange gange.	Brugeren får i applet besked om at adgang er spærret og han skal have midlertidig adgangskode for at blive låst op.	NemID er spærret og kan ikke bruges. For at få hjælp til dette, kan du kontakte NemID-support på tlf. 80 30 70 50	
OCES001	Brugeren har ikke OCES (fravalgt), men forsøger at logge ind på en OCES tjenesteudbyder.	Ingen fejlbesked i applet	Du har kun sagt ja til at bruge NemID til netbank. Ønsker du at bruge NemID til andre hjemmesider, skal du først tilknytte en offentlig digital signatur (OCES) til dit NemID - klik her www.nemid.nu [https://www.nemid.nu/privat/bestil_nemid/nemid_i_netbank/]	
OCES002	Brugeren har ikke OCES (ikke egnet), men forsøger at logge ind på en OCES tjenesteudbyder.		Ønsker du at bruge NemID til andet end netbank, skal du først tilknytte en offentlig digital signatur. Det gør du nemt og hurtigt ved at starte en ny bestilling af NemID på www.nemid.nu hvorved du får mulighed for at tilknytte en offentlig digital signatur - klik her www.nemid.nu. [https://www.nemid.nu/privat/bestil_nemid/]	
OCES003	Bruger har ikke POCES på dette device, men han har POCES på et andet		Der er ikke knyttet en digital signatur til det NemID, du har forsøgt at logge på med. Hvis du plejer at logge på [TU] med NemID, kan problemet	

Fejlkode	Beskrivelse	Tekst i applet	Tekstforslag til TU’er + evt. link til DCH	Anbefaling til TU
			skyldes, at du har flere forskellige NemID og at du nu har brugt et andet NemID, end du plejer.	
OCES004	Brugeren har ikke OCES (POCES uegnet (ikke CPR nummer eller under 15 år)), men forsøger at logge ind på en OCES tjenesteudbyder		Du kan kun bruge NemID til netbank.	
OCES005	Fejlkoden sendes når brugeren forsøger at lave oceslogin/signing i en situation, hvor CA'en skal udstede et certifikat, men af tekniske årsager ikke formår at gøre det.	Der opstod en fejl under oprettelse af OCES certifikat til NemID Prøv at logge på igen.		TU anbefales at lave en reload af appletten, så brugeren kan prøve igen. Hvis problemet fortsætter henvises til support
OCES006	Brugeren har kun inaktive/utilgængelige POCES (eller slet ingen) på alle OTP enheder.	Ingen fejlbesked i applet	Du har ikke i øjeblikket ikke en aktiv offentlig digital signatur (OCES-certifikat) til NemID. Det kan du få ved at starte en bestilling af NemID, hvorved du vil få mulighed for at vælge at bestille og tilknytte en offentlig digital signatur til dit nuværende NemID. Start bestillingen her: www.nemid.nu [https://www.nemid.nu/privat/bestil_nemid/]	
SRV006	Brugerens session med serveren er blevet tabt. Dette vil oftest skyldes at brugeren har været for længe om at logge ind, men kan også skyldes	ingenting	Tidsgrænse overskredet. Forsøg igen	

Fejlkode	Beskrivelse	Tekst i applet	Tekstforslag til TU'er + evt. link til DCH	Anbefaling til TU
	hacking eller server problemer.			
APP001 APP002 APP004 SRV001 SRV002 SRV003 SRV005	Diverse tekniske fejl som skyldes problemer hos TU	Sker ingenting i applet, kun returkode til TU	Der er opstået en teknisk fejl. Forsøg igen. Kontakt [TU], hvis problemet fortsætter.	
APP003 SRV004	Diverse tekniske fejl som skyldes teknik hos Nets DanID		Der er opstået en teknisk fejl. Kontakt NemID-support på tlf. 80 30 70 50	
APP005	Brugeren har valgt ikke at have tillid til certifikatet, der kan verificere signaturen på appletten.	Du skal godkende Nets DanIDs certifikat, før du kan logge på. Genstart din browser og godkend certifikatet. Har du brug for hjælp, kan du kontakte NemID-support på tlf: 80 30 70 50	Du skal godkende Nets DanIDs certifikat, før du kan logge på med NemID. Genstart din browser og godkend certifikatet næste gang du bliver spurgt. Har du brug for hjælp, kan du kontakte NemID-support på tlf. 80 30 70 50	

Fejlkode	**Beskrivelse**	**Tekst i applet**	**Tekstforslag til TU'er + evt. link til DCH**	**Anbefaling til TU**
AUTH001	NemID har haft en midlertidig adgangskode tilknyttet, men brugeren har indtastet for mange forkerte forsøg, så den er nu blevet spærret	Applet informerer brugeren om at adgangskoden er spærret	Din NemID er spærret. Kontakt venligst NemID-support på tlf. 80 30 70 50.	TU anbefales at henvise brugeren til at kontakte supporten.
AUTH004	Brugeren har forsøgt at logge ind, men hans NemID er spærret (tidslåst).	Brugeren får her kun besked på, at han ikke kan logge på.	Dit NemID er midlertidigt låst og du kan endnu ikke logge på. Du kan logge på igen når den 8 timers tidslås er ophævet.	TU anbefales at henvise brugeren til at kontakte supporten
AUTH005	Brugeren har forsøgt at logge ind, men hans NemID er spærret (permanent).	Brugeren får her kun besked på, at han ikke kan logge på.	Dit NemID er spærret. Kontakt venligst NemID-support på tlf. 80 30 70 50.	TU anbefales at henvise brugeren til at kontakte supporten.
AUTH006	"Ikke flere nøgler" - hvis brugeren har opbrugt sine nøglekort, uden at der er tilknyttet et nyt.	Fejlbesked i applet	Kontakt NemID-support på tlf. 80 30 70 50	TU anbefales at henvise brugeren til at kontakte supporten.
AUTH007	Brugerens NemID adgangskode tilbagekaldes på grund af for mange fejlslagne indtastninsforsøg. Denne fejlkode returneres, hvis brugeren forsøger at godkende et NemID, der	Ingenting – der returneres svar til TU	Kontakt NemID-support på tlf. 80 30 70 50	TU anbefales at henvise brugeren til at kontakte supporten.

Fejlkode	Beskrivelse	Tekst i applet	Tekstforslag til TU'er + evt. link til DCH	Anbefaling til TU
	har været inddraget under en tidligere session.			
AUTH008	Brugerens NemID er ikke aktiveret, og brugeren har ikke en aktiv midlertidig adgangskode.	Ingenting – der returneres svar til TU	Kontakt NemID-support på tlf. 80 30 70 50	TU anbefales at henvise brugeren til at kontakte supporten.

4 Signering med Applet med OTP

Dette afsnit beskriver signeringsfunktionaliteten i Applet med OTP.

Der understøttes signering af almindelig tekst, HTML, XML og PDF dokumenter.

Hvis en signering mislykkes sendes en fejlkode tilbage til tjenesteudbyder. I modsætning til autentificering flow ved log-in, er fejlkoden ikke indpakket i et besked-format.

4.1 Almindelig tekst signering

Den tekst der ønskes signeret skal gives som en base64-kodet parameter til appletten. Ved succesfuld signering returneres den oprindelige base64streng uændret i det underskrevne dokument.

Klartekst signeringen vil vise teksten direkte for brugeren med hvide mellemrum bevaret.

Tekst signering specificeres med følgende applet parametre

```
<param name="ZIP_FILE_ALIAS" value="OpenSign2">
<param name="signtext" value="b3Blbm9jZXM=">
<param name="signTextFormat" value="PLAIN">
```

Parameteren “`ZIP_FILE_ALIAS`” skal angives til “`OpenSign2`”.

Parameteren ”`signText`” angiver signeringsteksten base64-kodet.

Parameteren ”`signTextFormat`” angiver at der ønskes signering af almindelig tekst.

4.2 HTML signering

HTML-formatet, der kan anvendes til signeringen af dokumenter udgør en delmængde af HTML 3.2 standarden [HTML32] med den ændring, at dokumentet skal være gyldig XML og ikke som standarden lejlighedsvis tillader, at der kan anvendes uafsluttede tags.

Nedenstående tabel viser de understøttede HTML-elementer. Elementerne og deres attributter, er som beskrevet i `[HTML32]`, undtagen for tilføjede attributter, der er markeret med en understregning.

HTML Element	Understøttede attributter
html	
body	text bgcolor class style
head	
style	Type
title	
p	align bgcolor style class
div	align bgcolor style class
ul	style class
ol	start type style class
li	class style
h1 h2 h3 h4 h5 h6	class style
font	face size color
table	border cellspacing cellpadding width align
tr	bgcolor class style
th td	bgcolor rowspan colspan align valign width class style
i b u	
center	
a	href name

HTML signering specificeres med følgende applet parametre

```
<param name="ZIP_FILE_ALIAS" value="OpenSign2">
<param name="signtext" value="PGgxPm9wZW5vY2VzPC9oMT4=">
<param name="signTextFormat" value="HTML">
```

Parametren “`ZIP_FILE_ALIAS`” skal angives til “`OpenSign2`” .

Parametren “`signText`” angiver signering HTML base64-kodet.

`Parametren ”signTextFormat”` angiver at der ønskes signering af HTML.

Signerede tekster skal altid gives som en base64. Links kan pege på navngivne ankre i dokumentet, men kan ikke pege på eksterne dokumenter.

CSS kan anvendes ved hjælp af `style`element eller attributterne`style` og `class`.Javadoc siden for `javax.swing.text.html.CSS` klassen forklarer i detaljer understøttelsen for CSS i Java.

4.3 XML signering

Underskrivelsen af XML-dokumenter tilføjer følgende parametre til appletten.

signTransformation
Et base64 kodet XSL-stylesheet transformationsark, der kan transformere XML-dokumentet i signText parameter til et HTML-dokument, der følger de regler, der er skitseret i afsnit 4.

signTransformationId
Denne valgfri parameter gør det muligt for udbyderen at tilføje en parameter, der beskriver den anvendte transformation. Værdien af denne parameter vil blive medtaget i det underskrevne dokument, og kan bistå tjenesteyderen i at kunne identificerer den transformation, der blev brugt til at vise den signerede tekst.

Parameteren bliver XML-encoded, inden den tilføjes det signerede dokument, men vil ikke blive ændret på andre måder.

XML signering specificeres med følgende applet parametre

```
    <param name="ZIP_FILE_ALIAS" value="OpenSign2">
    <param name="sigText"
value="PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiIHN0YW5kYWxvbmU9InllcyI/Pg0KPD94bWwtc3R5bGVzaGVldCB0eXBlPSJ0
ZXh0L3hzbCIgaHJlZj0idHV0b3JpYWxzLnhzbCI/Pg0KPG9wZW5zaWduPg0KICA8eG1sc2lnbj4NCiAgICA8dGl0bGU+WE1MIHRvU2lnbiAxPC90aXR
sZT4NCiAgPC94bWxzaWduPg0KICA8eG1sc2lnbj4NCiAgICA8dGl0bGU+WE1MIHRvU2lnbiAyPC90aXRsZT4NCiAgPC94bWxzaWduPg0KPC9vcGVuc2
lnbj4NCg=="> 


    <param name="signTransformation"
value="PD94bWwgdmVyc2lvbj0iMS4wIj8+DQo8eHNsOnN0eWxlc2hlZXQgdmVyc2lvbj0iMS4wIiB4bWxuczp4c2w9Imh0dHA6Ly93d3cudzMub3Jn
LzE5OTkvWFNML1RyYW5zZm9ybSI+DQoNCjx4c2w6dGVtcGxhdGUgbWF0Y2g9Ii8iPg0KPGh0bWw+DQo8Ym9keT4NCjxoMj5PcGVuU2lnbjwvaDI+DQo
gIDx4c2w6YXBwbHktdGVtcGxhdGVzLz4NCjwvYm9keT4NCjwvaHRtbD4NCjwveHNsOnRlbXBsYXRlPg0KDQo8eHNsOnRlbXBsYXRlIG1hdGNoPSJ4bW
xzaWduIj4NCiAgPHhzbDp2YWx1ZS1vZiBzZWxlY3Q9InRpdGxlIi8+DQo8L3hzbDp0ZW1wbGF0ZT4NCg0KPC94c2w6c3R5bGVzaGVldD4=">

    <param name="signTextFormat" value="XML">
```

Parametren "`ZIP_FILE_ALIAS`" skal angives til "`OpenSign2`" .

Parametren "`signText`" angiver signering XML base64-kodet.

Parametren "`signTransformation`" angiver XSLT stylesheet base64-kodet.

Parametren "`signTextFormat`" angiver at der ønskes signering af XML

4.4 PDF signering

Der understøttes signering af en delmængde af PDF. Se afsnit 4.4.1 PDF whitelisting.

PDF signering specificeres af følgende applet parametre

```
 <param name="ZIP_FILE_ALIAS" value="OpenSign2">
<param name="signText.uri" value="/servlet/DisplayPdfSignText;jsessionid=me9ekvnzvhvuf0pw88ypdf1u"/>
 <param name="signText.hash.value" value="4yn2IP1vtWUB7hdpTbIEnhsmO5k="/>
 <param name="signText.hash.algorithm" value="sha2"/>
 <param name="signTextFormat" value="PDF"/>
```

Parametren "`ZIP_FILE_ALIAS`" skal angives til "`OpenSign2`" .

Parametren "`signText.uri`" angiver en lokation hvorfra det PDF dokument der ønskes signeret kan hentes. Bemærk at dokumentet hentes internt i appletten og ikke i den indlejrende browser. Det kan derfor være nødvendigt at medsende information om den aktuelle http session – f.eks. som angivet i eksemplet ovenover.

Parameteren ”`signText.hash.value`” angiver en hash værdi at det PDF dokument der ønskes signeret.

Parameteren ”`signText.hash.algorithm`” skal angives til ”sha2”

`Parameteren ”signTextFormat”` angiver at der ønskes signering af PDF

Når der specificeres PDF signering på ovenstående måde viser appletten umiddelbart PDF dokumentet i det primære signeringsvindue.

4.4.1 PDF whitelisting

Af sikkerhedsmæssige hensyn, og for at sikre, at et PDF dokument ikke ændrer indhold eller udseende, er det ikke hele PDF-specifikationen, der understøttes. Med andre ord er det ikke nødvendigvis alle PDF-dokumenter, der kan signeres. Der anvendes en såkaldt *whitelist*, som indeholder de elementer fra PDF-specifikationen, som understøttes – og dermed er tilladt i et PDF-dokument.

Der er udviklet et værktøj til at kontrollere at et PDF dokument kan bestå valideringen. Værktøjet hedder ”SignViewer”, og kan installeres lokalt og tilgås både programmatisk og via GUI.

Der kan være stor forskel på hvordan forskellige værktøjer danner et PDF-dokument. Adobe gør det én måde, Microsoft på en anden osv.

Som udgangspunkt er det udelukkende en delmængde elementer fra Adobe PDF-specifikationen, som er understøttet. Der er dog en enkelt undtagelse; et antal elementer, som bruges af Microsoft Office, og som ikke er en del af PDF-specifikationen, er også inkluderet i whitelisten.

Det er følgende Microsoft Office elementer, som er understøttet:

- /Workbook
- /Textbox
- /Endnote
- /Worksheet
- /Macrosheet
- /Annotation
- /Dialogsheet
- /Chartsheet

- /Diagram
- /Footnote
- /Chart
- /Slide
- /InlineShape
- /Artifact
- /Figure
- /Formula
- /Link

Fonte i dokumentet skal som udgangspunkt være indlejret i dokumentet. Dog er nedenstående fonte direkte understøttet, og derfor ikke nødvendige at indlejre:

- Times-Roman
- Times-Bold
- Times-Italic
- Times-BoldItalic
- Helvetica
- Helvetica-Bold
- Helvetica-Oblique
- Helvetica-BoldOblique
- Courier
- Courier-Bold
- Courier-Oblique
- Courier-BoldOblique
- Symbol
- ZapfDingbats

Følgende er eksempler på PDF-elementer, som ikke er understøttet.

OpenAction

Dette element kan bruges til at udføre en handling når et PDF-dokument åbnes. Det kan f.eks. være at afvikle et eksternt program, vise en pop-up m.m. Dette indebærer en sikkerhedsrisiko ligesom forskellige PDF-visere kan vise dokumentet på forskellige måder, og der dermed er risiko for at det signerede PDF-dokument præsenteres anderledes end i den signerede udgave.

EmbeddedFile / FileAttachment

Indebærer en sikkerhedsrisiko, da eventuelle skadelige filer kan pakkes i et PDF-dokument.

Sound / Movie / Widget

Signerings-appletten understøtter ikke disse elementer, og derfor er de ikke tilladt.

GoToR / GoTo E

Disse elementer giver mulighed for at åbne et eksternt PDF-dokument, og er derfor ikke tilladt. Al nødvendig information skal være i det samme PDF-dokument.

SubmitForm / AcroForm

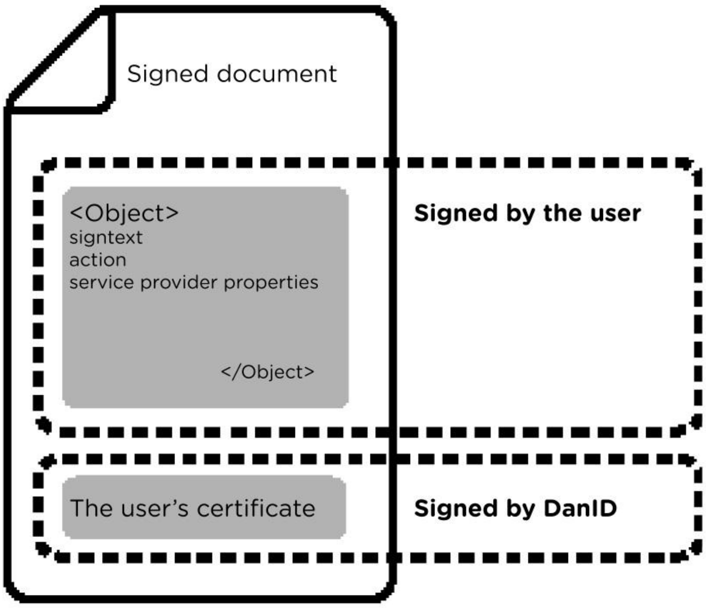
Et signeret PDF-dokuments indhold kan ikke ændres, og derfor tillades formulerer ikke.

Den komplette whitelist er beskrevet i Appendiks A – PDF Whitelist.

4.5 Signerede dokumenter

Dette afsnit beskriver opbygningen af et succesfuldt signeret dokument.

Strukturen af et signeret dokument er skitseret i følgende figur.



Figur 2 - Opbygning af et signeret dokument.

Et signeret dokument består af tre hovedkomponenter.

- Et objekt element i besiddelse af en række attributter. Hver attribut indeholder en markering med angivelse af, om attributten var synlig under signering.
- En signatur på objektet. Bemærk, at signaturen dækker hele objektet, og dermed samtlige attributter.
- Et certifikat, der er i stand til at verificere signaturen.

4.5.1 Properties

Dette afsnit indeholder oplysninger om de properties, et signeret dokument kan indeholde.

signtext
Denne property indeholder den identiske tekst til den, der oprindeligt blev givet som input til appletten i `signtext` parameteren af tjenesteudbyderen.

stylesheetDigest
Hvis signeringsteksten var et XML-dokument, vil denne property indeholde et SHA-256 digest af det stylesheet, som blev brugt til XML-transformation. Denne property er kun medtaget i XML signeringsresultatet.

stylesheetIdentifier
Tjenesteudbyderens egen identifier for det stylesheet, der blev anvendt til XML-signeringen. Propertien er kun til stede, hvis `SignTransformationId` parameteren oprindeligt blev givet til appletten. Indholdet er XML-encoded, inden den kopieres ind i denne property.

action
Er medtaget for at opretholde bagudkompatibilitet med OCES I. Den vil altid indeholde værdien "sign".
Udover de ovennævnte properties, vil det signerede dokument også indeholde properties, der kan have været angivet af tjenesteudbyderen ved hjælp af applet parameteren `SignProperties`.

4.6 Signering med attachments

Dette afsnit beskriver, hvorledes man kan foretage signering med attachments vha. signeringsappletten.

4.6.1 Attachments-parameter

Hvis der i forbindelse med signering skal være attachments, skal disse medgives som parameteren ”attachments” og værdien af denne skal være en base64-indkodning af et XML-dokument, der beskriver de attachments, der skal signeres. Parameteren indsættes på samme måde som signeringsteksten.

4.6.2 Udformning af XML

Rod-elementet i XML-dokumentet er ”<attachments>” og dette indeholder et antal ”<attachment>”-elementer, der hver beskriver et attachment. Hvert attachment beskrives af titel, sti, type, størrelse og hashværdi. Desuden kan der være et ”<optional>”-element, der angiver, at det er valgfrit, om dette attachment indgår i signeringen.

For at få den rette hashværdi skal man først beregne SHA256 på attachment-filen og resultatet af dette skal herefter base64-encodes.

Stien, der skal knyttes til et attachment, er relativ til roden af applikationen, dvs. at hvis appletten er på https://bank.dk/app/abc/longtermSign.jsp og et attachment ligger på https://bank.dk/app/abc/example.txt, skal stien være app/abc/example.txt.

Eksempel på XML følger længere nede.

4.6.3 MIME-typer

Når signeringen skal foregå, kan man se et vindue, der viser alle attachments med bl.a. titel. Her kan man gemme attachments og visse typer kan man også få vist indholdet af. Man kan have attachments af mange forskellige typer, men det er kun attachments af flg. Typer der kan vises direkte: text/plain, image/gif, text/html, text/rtf og application/pdf (kræver at pdf plugin er installeret).

4.6.4 Samlet størrelse

Der er en maksimal størrelse på det samlede, der skal signeres, dvs. størrelsen af signeringsteksten samt attachments. Denne er knap 1 MB så længe der anvendes JavaScript til overførelse af signaturen. Dette gælder i øvrigt også for signeringsteksten alene, såfremt der ikke er attachments.

Hvis den samlede størrelse overskrider 1MB kan der sættes følgende applet parameter

```
<param name="signText.chunk" value="true">
```

Samtidig skal der inkluderes følgende ekstra javascript i html siden

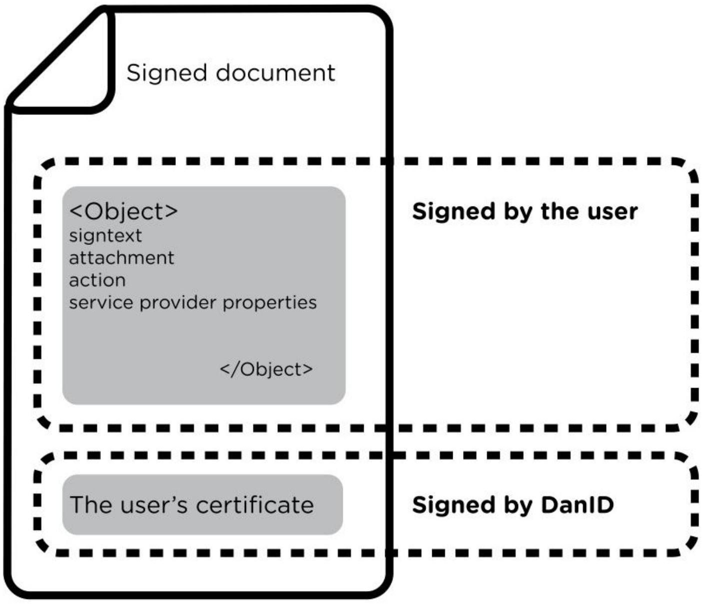
```
var signatureChunk;

function addChunk(chunk) {
  signatureChunk = signatureChunk + chunk;
}

function allChunk() {
  onSignOK(signatureChunk);
}
```

OTP- og OpenSign applet supporterer en samlet størrelse på signeringstekst og bilag med denne funktionalitet på op til 10 MB.

4.6.5 Signaturen

Den færdige signatur kommer til at indeholde en del, hvor hver attachment forefindes med attributter samt en base64-indkodning af indholdet. Det er endnu en property udover signtext, stylesheetdigest osv.

På nedenstående figur ses det at ”attachment” figurerer efter ”signtext”.



Figur 3 - Opbygning af et signeret dokument med attachment.

4.6.6 Eksempel

```
<?xml version="1.0" encoding="UTF-8" ?>
<attachments>

<attachment>
<title>A sample png sample</title>
<path>/sp/demo/attachments/sample.png</path>
<mimeType>image/png</mimeType>
<size>716</size>
<hashValue>a1zoiDHyz3vugBcjyRz06X77rwjWQD1Vg4eFC7+gy2Q=</hash
Value>
</attachment>

<attachment>
<title>A sample gif image</title>
```

```
<path>/sp/demo/attachments/sample.gif</path>
<mimeType>image/gif</mimeType>
<size>473</size>
<hashValue>yk0BVPVlPwFfN4Z/pReCvcoFMipKP3VzU8dJHPOdqbM=</hash
Value>
</attachment>

<attachment>
<title>A sample text document (optional)</title>
<path>/sp/demo/attachments/sample.txt</path>
<mimeType>text/plain</mimeType>
<size>5</size>
<hashValue>jgUeaO9PNjH2X211MYJxE82UR7FsE52wdaiGspp+Bcc=</hash
Value>
<optional/>
</attachment>

<attachment>
<title>A sample html document</title>
<path>/sp/demo/attachments/sample.html</path>
<mimeType>text/html</mimeType>
<size>94</size>
<hashValue>aVk8a8xH/nWMJanM1dttOEY59eg75BNmK2xtDLXB2jo=</hash
Value>
</attachment>

<attachment>
<title>A sample rtf document</title>
<path>/sp/demo/attachments/sample.rtf</path>
<mimeType>text/rtf</mimeType>
<size>31896</size>
<hashValue>EerJtv5Ay2ohvPufBB76TmZX/+SgU3F8/iW66rhKMNU=</hash
Value>
</attachment>

<attachment>
<title>A sample pdf document (optional, broken)</title>
<path>/sp/demo/attachments/sample.pdf</path>
<mimeType>text/pdf</mimeType>
<size>2610</size>
<hashValue>/wofJ78MfpjYHtKc/tCQ/ekrCOPGajpe6pnonFCENZE0</hash
Value>
<optional/>
</attachment>

</attachments>
```

5 Integration til Applet uden OTP

Dette er den eksisterende OpenOCES-applet. Fra version 1.8.0 er det muligt at benytte denne applet til både OCES I og OCES II log-in og signering. Applet uden OTP benyttes ligeledes til login med NemId på Hardware.

Bemærk at Applet uden OTP ikke benyttes af brugere med nøglekort (OTP) eller nøgleviser. Applet uden OTP loades ligesom Applet med OTP fra Nets DanID.

Et eksempel på hvordan appletten loades kunne være:

```
<applet id="signing_applet" name="signing_applet"
 code="org.openoces.opensign.client.applet.bootstrap.BootApplet" width="440"
  height="100" codebase="https://opensign.danid.dk/" archive="OpenSign-bootstrapped.jar"
  mayscript="true" alt="Opensign applet">

<param name="ZIP_FILE_ALIAS" value="OpenLogon" >
<param name="ZIP_BASE_URL" value="https://opensign.danid.dk/plugins" >
<param name="MS_SUPPORT" value="bcjce" >
<param name="SUN_SUPPORT" value="jsse" >
<param name="STRIP_ZIP" value="yes" >
<param name="EXTRA_ZIP_FILE_NAMES" value="capi,pkcs12,cdcard,oces" >
<param name="LOG_LEVEL" value="INFO" >
<param name="locale" value="da,DK" >

<param name="cabbase" value="https://opensign.danid.dk/OpenSign-bootstrapped.cab" />

<param name="key.store.directory" value="null" />
<param name="loglevel" value="info" />
<param name="background" value="255,255,255" />
<param name="socialsecuritynumber" value="no" />
<param name="optionalid" value="no" />
<param name="opensign.doappletrequest" value="false" />
<param name="opensign.doappletrequestonmac" value="false" />
<param name="logonto" value="www.nets-danid.dk" />
<param name="cdkortservice" value="demo" />
<param name="signproperties" value="challenge=<CHALLENGE>" />
<param name="subjectdnfilter" value="" />
<param name="issuerdnfilter"  value="" />
<param name="opensign.message.name" value="message" />
<param name="opensign.result.name" value="result" />
<param name="gui" value="modern" />
OpenSign applet
</applet>
```

Som det kan ses loades appletten i ovenstående eksempel fra: https://opensign.danid.dk

OBS: Debug flag ikke bør anvendes ved normal drift.

5.1 Parametre

Dette dokument vil ikke beskrive alle parametre. Så for en nærmere beskrivelse henvises til den almindelige, gældende vejledning, der kan findes på www.openoces.org.

Signproperties
Denne parameter indeholder egenskaber som skal signes. Den bliver blandt andet benyttet til at angive challenge.

subjectdnfilter
Denne parameter kan bruges på Windows-platformen til kun at vise bestemte certifikater fra brugerens certificatstore. Hvis man angiver:

<param name="subjectdnfilter" value="UElEOg=="/>

…Vil kun de POCES certifikater, som ligger i certificatstore blive vist.

Værdien"UElEOg==” er base64-kodning af ”PID:”, og kun POCES-certifikater har ”PID:” som delstreng i subjectdn. Hvis man i stedet angiver ”UklEOg==”, som er base64-kodning af ”RID:”, vises kun MOCES-certifikaterne i certificatstore.

Hvis man ønsker muligheden for både at logge ind med både MOCES og POCES, bør subjectdnfilter være tom, således at både certifikater med henholdsvis ”RID:” og ”PID:”i subjectdn i certificatstore, vil blive vist i dropdown boksen i OpenSign.

6 Validering af certifikat

Sikkerhedspakken gør brug af OOAPI (OpenOCES API) på en måde, så det er nemt at implementere et typisk log-in og signering.

6.1 Processen ved validering

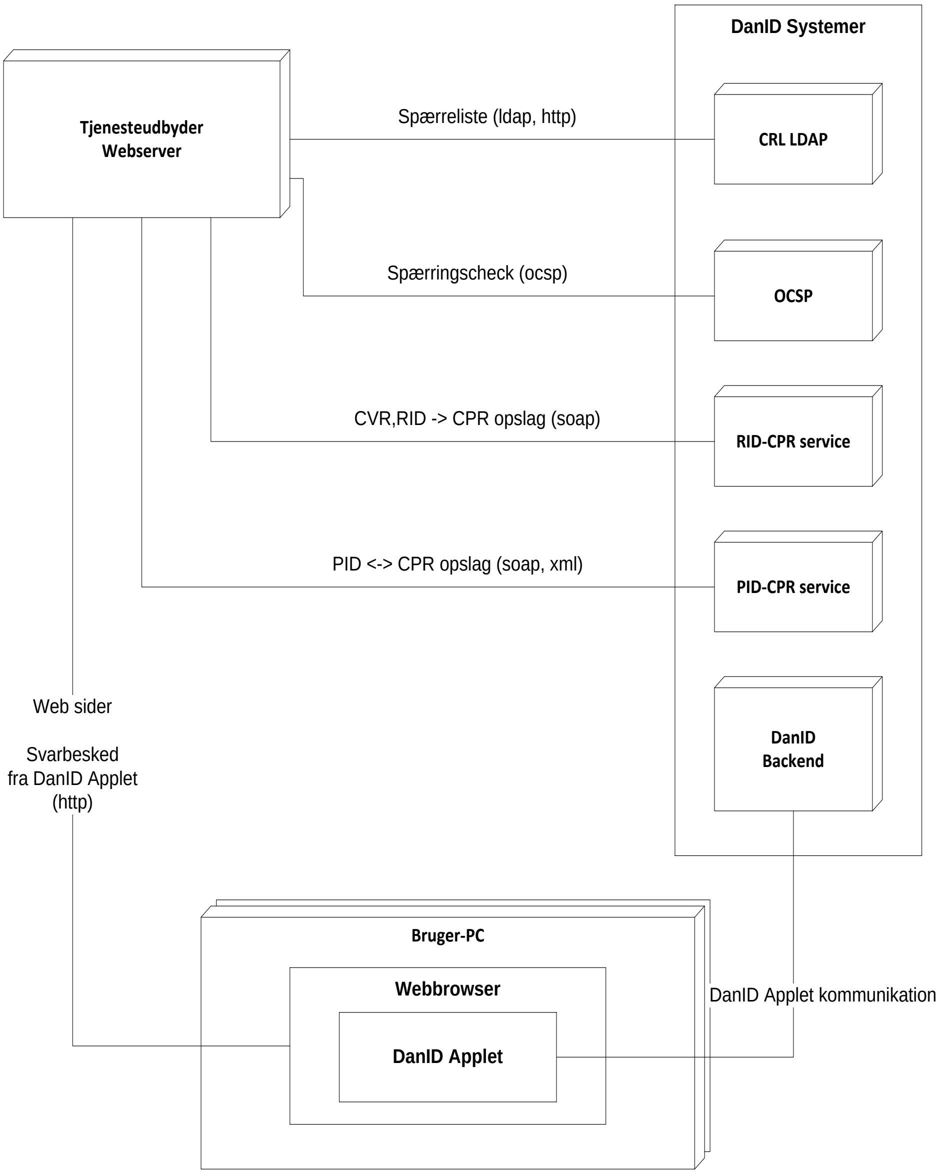
Når brugeren har autentificeret sig, sendes signaturen tilbage til webserveren pakket ind i XMLDSig. Appletten placerer XMLDSig-dokumentet i en HTML-formular på websiden, som appletten efterfølgende submitter. XMLDSig-dokumentet skal ekstraheres fra form submission på webserveren, og her vil der være et stykke referencekode, for hvordan signaturen i XMLDSig skal valideres.

Tjenesteudbyderen skal herefter have valideret certifikatet og eventuelt oversætte PID/RID-nummeret til et cpr.nr. Dette forudsætter, at tjenesteudbyderen kommer igennem følgende punkter, hvis der er tale om logon. Listen vil være tilsvarende for at verificere en signering:

1. Validere signaturen på XMLDSig
2. Trække certifikatet ud af XMLDSig
3. Validere certifikatet og identificere CA som OCES I eller OCES II gennem hele certifikatkæden til rodcertifikatet
4. Kontrollere at certifikatet ikke er udløbet
5. Kontrollere at certifikatet ikke er spærret
6. Trække PID eller RID ud af certifikatet
7. Oversæt PID eller RID til et cpr.nr.

Telefonisk verifikation af rodcertifkat. Korrektheden af rodcertifikatet kan verificeres, ved at sammenligne den miniatureudskrift (fingerprint), der fremgår af rodcertifikatet, med den miniatureudskrift som oplæses i telefonen ved opringning til tlf. 80 30 70 12.

Punkterne ovenfor afhænger af, om I vælger Nets DanIDs OOAPI til integrationen eller selv vil tilpasse/udbygge integrationen.



Figur 4 – Deploymentdiagram - Systemer der kommunikerer ved login.

Figur 4 viser hvilke systemer, der kan kommunikere i en login situation. I parentes er angivet hvilke protokoller, der kan anvendes. Selve login kommunikationen foregår som beskrevet i Figur 1. Herefter kan tjenesteudbyderen anvende Nets DanID's OOAPI til at gennemgå punkt 1 til 7 i førnævnte liste. Spærringschecket kan både udføres ved at hente en spærreliste (CRL) fra systemet CRL LDAP, eller ved at udføre et check mod Nets DanIDs OCSP (Online Certificate Status Protocol) system. Endelig kan RID-CPR servicen anvendes til at slå CPR-nummer op ud fra CVR-nummer og RID. Tilsvarende kan PID-CPR servicen anvendes til at slå cpr-nummer op ud fra PID, eller til at matche PID med cpr-nummer.

Læs mere om den direkte integration i Afsnit 7 - **Direkte integration til Nets DanIDs infrastruktur**.

6.2 Nets DanIDs tjenesteudbyderpakke

Formålet med Nets DanIDs tjenesteudbyderpakke er, at gøre det let for tjenesteudbydere at implementere NemID på deres hjemmeside.

6.2.1 Tjenesteudbyderpakkens ressourcer

Tjenesteudbyderpakken består foruden dokumentationspakken af følgende komponenter:

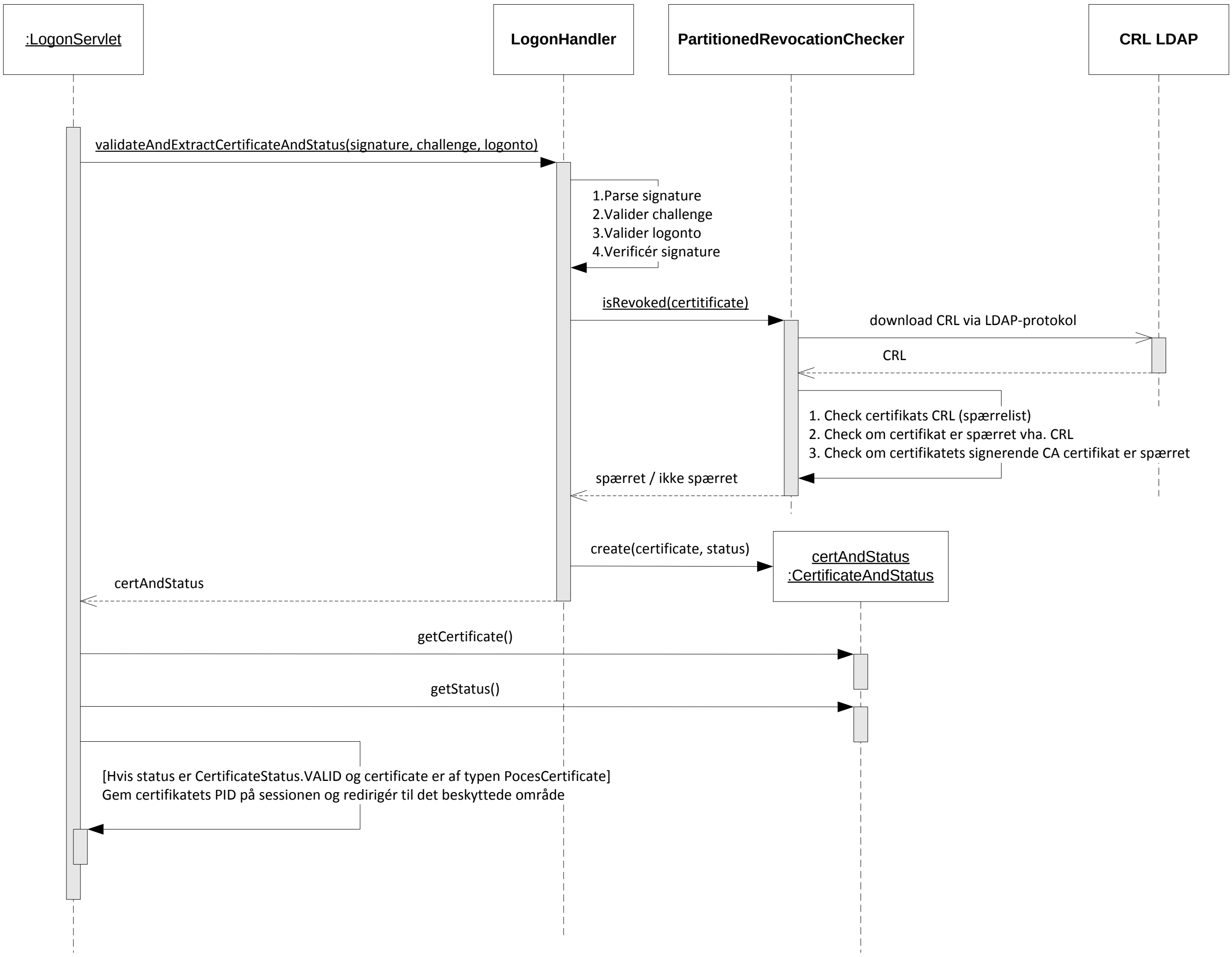
- **Tuexample-source.zip** – indeholder både test applet med OTP og eksempel på implementering af NemID (source format)
- **tuexample.war** - ovenstående indhold pakket i fil til Appachetomcat webserver (distribution af TU example)
- **ooapi-<version>-source.zip** – source udgave af OOAPI
- **ooapi-<version>.jar -** Kompileret udgave af OOAPI
- **ooapi-<version>-with-dependencies.jar -** Kompileret udgave af OOAPI inklusiv de jar-filer som OOAPI afhænger af.
- **ooapi.net-<version>.zip**. - .Net version af OOAPI
- **ooapi.net-<version>-souce.zip**. - Kildekode til OOAPI i .Net samt eksempel på implementering af NemID (source version).
- **Javadoc - tuexample.zip**. Java doc til TU Example
- **Javadoc – ooapi og sikkerhedspakke-<version>.zip**. Java doc til OOAPI og sikkerhedspakke

6.2.2 LogonHandler

En central del af OOAPI og sikkerhedspakken er klassen LogonHandler. Denne klasse tilbyder en metode til at få valideret de login data, OpenOcesLogon-appletten sender, og derudover at få returneret PID for den person, der er logget ind (RID hvis det er en medarbejdersignatur).

En vellykket validering af login data (dvs. et kald, som ikke har affødt nogle exceptions) sikrer følgende:

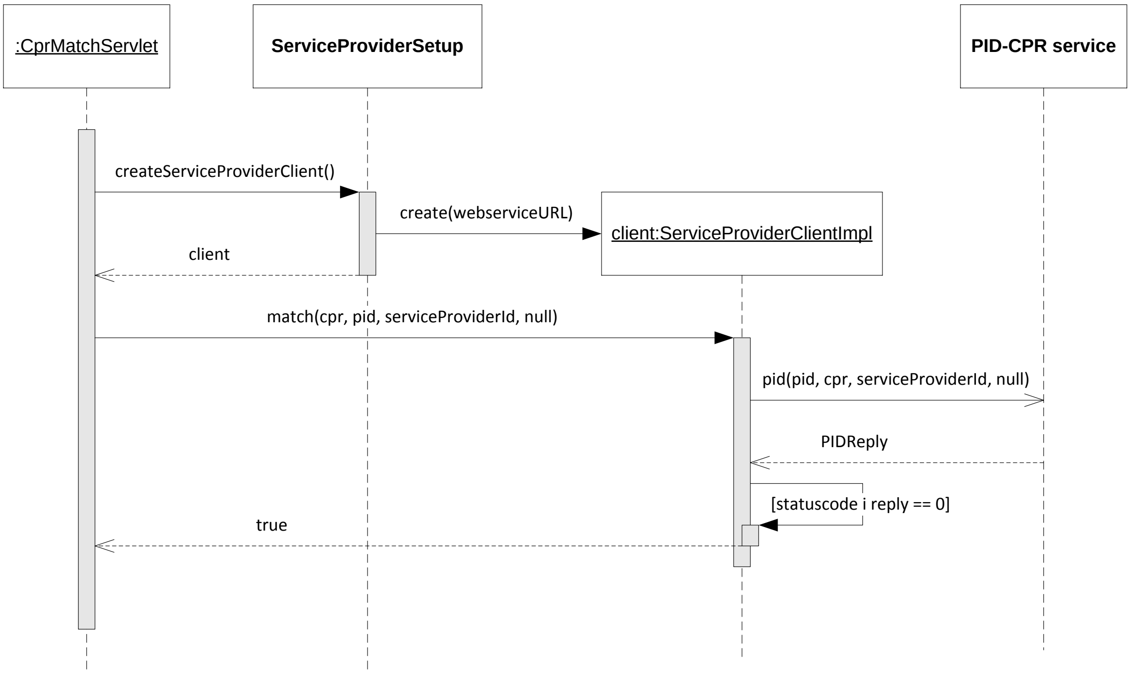
- at den signatur, som login data indeholder, er gyldig.
- at det certifikat, som login data indeholder, er gyldigt, og derfor ikke er udløbet eller spærret. Ved at undersøge returneret status kan man se hvilken tilstand certifikatet er i.



Figur 5 - Sekvensdiagram - LogonServlet håndterer login.

Figur ***5*** viser hvordan LogonServlet fra Tuexample-source.zip anvender LogonHandler til at validere et login og herunder hvordan LogonHandler validerer signatur, challenge og logonto parametrene. Challenge og logonto parametrene valideres ifht. tilsvarende parametre i signaturen.

LogonHandler parser signaturen og udtrækker herfra certifikat-kæden. Certifikat-kæden består af rod-CA'ens certifikat, den udstedende CA's certifikat og bruger-certifikatet. Denne certifikat-kæde verificeres ved at sikre at det udstedende CA's certifikat har signeret bruger-certifikatet og tilsvarende at rod-CA'ens certifikat har signeret den udstedende CA's certifikat. Herefter anvender LogonHandler klassen PartitionedRevocationChecker til at checke om bruger-certifikatet er spærret. PartionedRevocationChecker kigger i bruger-certifikatet for at identificere hvilken partiel spærreliste (CRL), certifikatet tilhører og henter herefter denne via CRL LDAP. Optræder bruger-certifikatet på den partielle spærreliste er certifikatet spærret.



Figur 6 - Sekvensdiagram - CprMatchServlet laver et vellykket match opslag.

Som en del af login-proceduren hos en tjenesteudbyder kan man bede brugeren om at indtaste sit cpr-nummer. Tjenesteudbyderen kan så bruge PID-CPR servicen til at validere at cpr-nummer og login PID matcher vha. af et match-opslag. *Figur* ***6*** viser hvordan CprMatchServlet udfører sådan et opslag vha. ServiceProviderClient. ServiceProviderClient er en webservice klient i OOAPI. Den kan anvendes til at kalde PID-CPR tjenesten. CprMatchServlet anvender

ServiceProviderSetup til at instantiere en ServiceProviderClient. ServiceProviderSetup kan desuden anvendes til at styre hvilket miljø (Environment) der afvikles imod samt konfigurere hvilken spærringschecker der skal anvendes. Der er findes tre spærringscheckere i Nets DanIDs OOAPI:

- PartitionedCrlRevocationChecker – Udfører spærringscheck ved at hente en partiel spærreliste fra CRL LDAP. Certifikatet angiver hvilken partiel spærreliste det tilhører.

- FullCrlRevocationChecker – Udfører spærringscheck ved at hente den fulde spærreliste over http. FullCrlRevocationChecker sørger for at holde den fulde spærreliste i en cache. Man kan styre levetid i cachen ved at sætte propertien ”crl.cache.timeout.http” i filen ooapi.properties.

- OCSPCertificateRevocationChecker – Udfører spærringscheck ved at kontakte Nets DanIDs OCSP system.

6.2.3 SignHandler

Klassen SignHandler tilbyder en metode til at validere output fra OpenSign-appletten mod en given aftaletekst. En vellykket validering af signeringsdata (dvs. et kald som returnerer *true*) sikrer følgende:

- at den signatur, som signeringsdata indeholder, er gyldig

- at det certifikat, som signeringsdata indeholder, er gyldigt, og derfor ikke er udløbet eller spærret. Ved at undersøge returneret status kan man se hvilken tilstand certifikatet er i.

6.2.4 Eksempel på webapplikation i java

Som eksempel på, hvordan OOAPI bruges, har Nets DanID udviklet en simpel webapplikation til at håndtere log-in og signering ved hjælp af NemID. Applikationen er bygget op om tre scenarier:

- NemID Privat (variant 1)

- NemID Erhverv (variant 2)

- NemID Privat og Erhverv (variant 3)

Hver af disse scenarier indeholder eksempler på log-in og signering med nøglekort, med OCES-II-nøglefil og med Digital Signatur-nøglefil.

Webapplikationen indeholder desuden et layout (interaktionsdesign), som Nets DanID anbefaler til inkorporering logon- og signeringsapplets.

Nedenfor finder du en kort introduktion til applikationens opbygning. Der henvises til Javadoc og kildekoden for yderligere information.

For at kunne kompilere applikationen, skal du som minimum have Maven version 2.1.1 installeret.

Applikationens opbygning

Applikationen er opbygget således:

tuexample	
src/main/java	Javaklasser
src/main/resources	Andre programressourcer
src/main/webapp	Web-filer
resources	Stylesheets og Javascript
variant1	Scenariet med NemID Privat
variant2	Scenariet med NemID Erhverv
variant3	Scenariet med NemID Privat og Erhverv
extras	Diverse opsætning og PID-opslag
WEB-INF/web.xml	Webapplikationens konfiguration
*.jsp	Websider
pom.xml	Maven-POM

Idéen er at hvis du kun er interesseret i variant 2, kan du se helt bort fra indholdet i variant1- og variant3-mapperne.

Generering af Javadoc

Hvis du vil generere Javadoc, skal du køre kommandoen:

```
mvn javadoc:javadoc
```

fra kommandolinjen.

Den genererede Javadoc findes i **target/site/apidocs**.

Eksekvering med Jetty

Hvis du vil benytte Jetty til at eksekvere applikationen, skal du køre kommandoen

```
mvn jetty:run
```

fra kommandolinjen.

Derefter kan applikationen findes på http://localhost:8082/tuexample.

Eksekvering med Tomcat

Hvis du vil benytte en præinstalleret Tomcat til at eksekvere applikationen, skal du kørekommandoen

```
mvn install
```

fra kommandolinjen.

Applikationen pakkes derved som et *war*-arkiv.

Kopiér dernæst **target/tuexample.war** til **<tomcat-dir>/webapps**.

Herefter kan applikationen findes på http://localhost:8080/tuexample (eller anden port, afhængig af hvorledes Tomcat er konfigureret.)

Eksempel på log-in

Under variant1, variant2 og variant3 ligger der et underkatalog der hedder ”restricted”. Indhold i dette katalog kan først tilgås efter et log-in. Hvis du forsøger at tilgå en side herunder, uden at være logget ind, vil du af LogonFilter blive dirigeret videre til loginsiden. Her har du mulighed for at logge ind med NemID eller Digital Signatur.

Videredirigeringen er sat op i web.xml på følgende måde (for variant 1):

```
<filter>
  <filter-name>variant1SecurityFilter</filter-name>
  <filter-class>dk.certifikat.tuexample.variant1.LogonFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>variant1SecurityFilter</filter-name>
  <url-pattern>/variant1/restricted/*</url-pattern>
</filter-mapping>
```

Selve LogonFilter-klassen udgøres af AbstractLogonFilter (der er generel for de tre scenarier) og en subklasse i hver variant-pakke. AbstractLogonFilter sørger for den overordnede kontrol med login-checket, mens hver subklasse sørger for at redirigere til hver sin login-side. Desuden checker hver subklasse for at der logges ind med et certifikat af den forventede type.

Selve login-siden består af flere faner, fx til login med nøglekort eller Digital Signatur. Som udgangspunkt sendes brugeren til fanen hvor man kan logge ind med nøglekort, men på hver fane er det muligt at sætte en cookie, så systemet husker hvilken fane man ønsker at bruge til login. Derfor kigger AbstractLogonFilter på cookien ”preferredLogin” for at afgøre hvilken fane brugeren skal redirigeres til.

Uanset hvilken login-metode brugeren vælger, ender appletten, når brugeren klikker på **OK**, med at sende resultatet til LogonServlet. LogonServlet er sat op i web.xml således (for variant 1):

```
<servlet>
  <servlet-name>variant1LogonServlet</servlet-name>
  <servlet-class>dk.certifikat.tuexample.variant1.LogonServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>variant1LogonServlet</servlet-name>
  <url-pattern>/variant1/logon.html</url-pattern>
</servlet-mapping>
```

Præcis som med LogonFilter-klassen udgøres LogonServlet af AbstractLogonServlet (der er generel for de tre scenarier) og en subklasse i hver variant-pakke.

Det er LogonServlet'ens opgave at checke at der er logget ind med et gyldigt certifikat, samt om certifikatet er af den forventede type.

LogonServlet-servletten bruger sikkerhedspakkens LogonHandler til at få valideret logindata og til at få udtrukket et PID- eller RID-nummer fra logindata. Når LogonHandler-klassen har returneret et PID- eller RID-nummer, gemmes dette i sessionen, som her i variant 1:

```
@Override
protected void logon(HttpServletRequest req, HttpServletResponse resp, CertificateAndStatus
certificateAndStatus) throws IOException, ServletException {
    HttpSession httpSession = req.getSession();
    if (isPoces(certificateAndStatus)) {
        String pid = ((PocesCertificate) certificateAndStatus.getCertificate()).getPid();
        httpSession.setAttribute(KEY_PID, pid);
        httpSession.setAttribute(KEY_LOGGED_IN, Boolean.TRUE);
        resp.sendRedirect(req.getContextPath()+"/variant1/restricted/kvittering.jsp");
    } else {
        …
    }
}
```

Eksemplet på en webapplikation er meget simpelt. I en rigtig webapplikation kan man forestille sig, at PID- eller RID-nummeret f.eks. bruges til at kontrollere,om brugeren med det angivne PID- eller RID-nummer har adgang til de beskyttede sider. Herefter kunne applikationen slå PID- eller RID-nummeret op i en lokal brugertabel og vise en tekst, der beskriver, hvem der er logget ind (f.eks. "Peter Madsen er logget ind").

URL'en *logout.html* rammer LogoutServlet-serveletten, som logger brugeren ud og sender ham videre til forsiden af det givne scenario. Brugeren logges ud ved at fjerne den nøgle fra HttpSession-objektet der fortæller at vedkommende er logget ind.

Eksempel på signering

Signeringen består, ligesom login, af flere faner. Her kan man fx signere med nøglekort eller Digital Signatur. Baseret på ”preferredLogin”-cookien (se under eksempel på log-in) vises den relevante side for brugeren.

SignServlet-servletten, der ligesom LogonFilter og LogonServlet består af en AbstractSignServlet-klasse og en scenario-specifik subklasse, modtager resultatet fra den givne applet og validerer at det benyttede certifikat er gyldigt, at certifikatet er af korrekt type, og slutteligt at det er den korrekte tekst der er signeret. For at checke sidstnævnte, sættes aftaleteksten som en attribut på sessionen.

SignServlet benytter SignHandler til at afgøre om signeringsdataene er gyldige. Afhængigt af resultatet sendes brugeren videre til enten en successide eller en fejlside.

6.2.5 Eksempel på webapplikation i .NET

Som eksempel på, hvordan OOAPI.NET bruges, har Nets DanID udviklet en simpel webapplikation til at håndtere log-in og signering ved hjælp af NemID.

Applikationen er bygget op om tre scenarier:

- NemID Privat (variant 1)
- NemID Erhverv (variant 2)
- NemID Privat og Erhverv (variant 3)

Hver af disse scenarier indeholder eksempler på log-in og signering med nøglekort, med OCES-II-nøglefil og med Digital Signatur-nøglefil.

Webapplikationen indeholder desuden et layout (interaktionsdesign), som Nets DanID anbefaler til inkorporering logon- og signeringsapplets.

Nedenfor finder du en kort introduktion til applikationens opbygning. Der henvises til dokumentationsfiler og kildekode for yderligere information.

For at kunne kompilere applikationen skal du som minimum have .NET 3.5.

Applikationens opbygning

Applikationen er opbygget således:

Tuexample.net

include/	Filer som bliver inkluderet i html-siderne
variant1/	Scenariet med NemID Privat
variant2/	Scenariet med NemID Erhverv
variant3/	Scenariet med NemID Privat og Erhverv
extras/	Diverse opsætning og PID-opslag
Properties/	Indeholder assembly.cs
resources/	js, css og billedfiler
tuexample/	CS-filer til at lave challenge og fejlhåndtering
*.aspx	Websider
Web.config	Webapplikationenskonfiguration

Idéen er at hvis du kun er interesseret i variant 2, kan du se helt bort fra indholdet i variant1- og variant3-mapperne.

Generering af dokumentation

For generering af ndoc, kig i readme-filen:

`Ooapi.net/Docs/README.txt`

Opstart fra Visual Studio

Hvis du vil starte tuexample.net op i Visual Studio, så kig i readme-filen for vejledning:

`Ooapi.net/Docs/README.txt`

Opsætning af webapplikationen

I filen web.config findes opsætningen af webapplikationen. Under client ->endpoint findes opsætningen til pidserviceprovider. Som det kan bemærkes, er der angivet en adresse til PID/RID-CPR-tjenesten. Denne bliver dog overstyret når ooapi laver lookup, da ooapi selv indeholder en property som angiver URL'en på PID-tjenesten i et given miljø.

Under appSettings findes der nogle properties som blandt andet benyttes til at angive URL'en til Applet uden OTP (OpenSign) og URL'en til Applet med OTP. Disse properties anvendes også til at angive det Service Provider-ID (SPID) som man benytter til PID/RID-CPR-validering. Ligeledes angiver man også den pfx-fil, der skal benyttes til at signere appletparametre til NemID-appletten.

```
<appSettings>
<add key="openOcesLocation" value="https://erhverv.ig.certifikat.dk/applet/OpenSign/" />
<add key="openOcesJar" value="OpenSign-bootstrapped.jar" />
<add key="nemIDAppletPrefix" value="https://syst2.danid.dk"/>
<add key="pfxFile" value="C:\ooapi.net\ooapi.net\Docs\korrektCSPpfxname.pfx"/>
<add key="pfxPassword" value="Test1234"/>
<add key="spidPid" value="10005"/>
<add key="trustedEnvironment" value="OcesIiDanidEnvIgtest,OcesIDanidEnvSystemtest"/>
</appSettings>
```

Kig i readme-filen for mere information:

`Ooapi.net/Docs/README.txt`

6.2.6 Validering af CPR-numre

Klassen CprMatchServlet implementerer et eksempel på, hvordan man validerer sammenhængen mellem en brugers certifikat og CPR-nummer ved kald til PID/RID-CPR-tjenesten.

```
/* validerer sammenhæng mellem CPR-nummer og PID-nummer ved kald til PID/CPR-webservicen */
private boolean validateCPR(String cpr, String pid) {
  if (cpr != null) {
    ServiceProviderClientImpl pidCprService = ServiceProviderClientImpl.createForTestEnv();
    return pidCprService.match(cpr, pid, "44");
  } else {
    return false;
  }
}
```

Kald mod denne tjenste kræver, at man autentificerer sig over for den, dels med et virksomhedscertifikat (VOCES) og dels ved angivelse at et Service Provider-id (kaldet ”SPID”).

Service Provider-id’et (SPID) er det angivne "44" i koden ovenfor. For at etablere SSL-forbindelsen mod servicen kræves derudover, at du konfigurerer et *truststore*.

Konfigurationen af disse to angives i filen *pidclientsecurity.xml*, hvor *wsclientkeystore.jks* svarer til virksomhedscertifikatet, og *wsclienttruststore.jks* svarer til den nødvendige truststore.

Virksomhedscertifikatet (VOCES) er unikt for hver tjenesteudbyder, og *wsclienttruststore.jks* er statisk.

7 Viderestilling til nemid.nu med NemLog-in Single Sign On

Hvis man er oprettet som tjenesteudbyder ved NemLog-in, og man har en bruger, der er logget ind via NemLog-in, kan man viderestille en bruger direkte til selvbetjeningen på www.nemid.nu. Dette gøres ved at viderestille brugeren til https://www.nemid.nu/nemlogin.

8 Direkte integration til Nets DanIDs infrastruktur

Som tidligere nævnt anbefales det, at tjenesteudbydere tager udgangspunkt i Nets DanIDs tjenesteudbyderpakke og OOAPI ved implementering af NemID. Dette modul dækker de fleste integrationsmuligheder og vil i langt de fleste tilfælde være dækkende.

For de tjenesteudbydere, der selv udvikler deres interface til DanIDs infrastruktur, henvises til følgende dokumenter, der alle er en del af Tjenesteudbyderpakken:

- Introduktion til NemID og Tjenesteudbyderpakken
- Specifikationsdokument for servicen PID-CPR
    - En beskrivelse af servicen PID-CPR.
- Specifikationsdokument for servicen RID-CPR
    - En beskrivelse af servicen RID-CPR.
- Specifikationsdokument for LDAP API
    - En beskrivelse af, hvordan spærrelister kan hentes via LDAP og hvilke ting man skal være opmærksom på.
- Specifikationsdokument for OCSP
    - En beskrivelse af, hvordan man tilgår OCSP-responderen, og hvordan interfacet er.

9 Appendiks A – PDF Whitelist

Følgende er den komplette PDF whitelist:

PDF typer:

/FontDescriptor
/Page
/Font
/Metadata

PDF nøgler:

/Encoding
/ExtGState
/ColorSpace
/Pattern
/Shading
/XObject
/ProcSet
/Properties
/BaseFont
/Name
/Dests
/Dest
/Info
/Font

PDF navne:

/1.1
/1.2
/1.3
/1.4
/1.5
/1.6
/1.7
/2.2
/83pv-RKSJ-H
/90ms-RKSJ-H
/90ms-RKSJ-V
/90msp-RKSJ-H
/90msp-RKSJ-V
/90pv-RKSJ-H
/A
/A85
/AC
/ADBE
/AESV2
/AHx
/AIS
/AN
/AP
/AS
/ASCII85Decode
/ASCIIHexDecode
/AbsoluteColorimetric
/Accepted
/AccurateScreens
/Action
/ActualText
/Add-RKSJ-H
/Add-RKSJ-V
/AddRevInfo
/Adobe.PPKLite
/After
/All
/AllOff
/AllOn
/AllPages
/Alpha
/AlphaNum
/Alphabetic
/Alt
/Alternate
/AlternateImages
/AlternatePresentations
/Alternates
/Angle
/Annot
/AnnotStates
/Annotations
/Annots
/AntiAlias
/AnyOff
/AnyOn
/App
/AppDefault
/Approced
/Art
/ArtBox
/AsIs
/Ascent

/Attached
/Attestation
/AuthEvent
/Author
/Auto
/AvgWidth
/B
/B5pc-H
/B5pc-V
/BBox
/BC
/BE
/BG
/BG-EUC-H
/BG-EUC-V
/BG2
/BM
/BS
/Background
/BackgroundColor
/BarcodePlaintext
/BaseEncoding
/BaseFont
/BaseState
/BaseVersion
/BaselineShift
/Bead
/Before
/BibEntry
/BitsPerComponent
/BitsPerCoordinate
/BitsPerFlag
/BitsPerSample
/Black
/BlackPoint
/Blackls1
/BleedBox
/Block
/BlockAlign
/BlockQuote
/Blue
/Border
/BorderColor
/BorderStyle
/BorderThickness
/Both
/Bounds
/BoxColorInfo
/ByteRange
/C
/C0
/C1
/CA
/CCF
/CCITTFaxDecode
/CF
/CFM
/CICI.SignIt
/CIDFontType0
/CIDFontType0C
/CIDFontType2
/CIDInit
/CIDSet
/CIDSystemInfo
/CIDToGIDMap
/CMap
/CMapName
/CMapType
/CNS-EUC-H
/CNS-EUR-V
/CO
/CP
/CS
/CYX
/CalGray
/CalRGB
/Cancelled
/Cap
/CapHeight
/Caption
/Caret
/Catalog
/Center
/CenterWindow
/Cert
/Changes
/CharProcs
/CharSet
/Circle
/ClassMap
/Code
/ColSpan
/Collection
/CollectionField
/CollectionItem
/CollectionSort
/CollectionSubItem
/Color
/ColorBurn
/ColorDodge
/ColorSpace
/ColorTransform

```
/Colorants
/Colors
/Column
/ColumnCount
/ColumnGap
/ColumnWidth
/Columns
/Comment
/Completed
/Components
/Confidential
/Configs
/ContactInfo
/Content
/Contents
/Coords
/Copy
/CosineDot
/Count
/Courier
/Courier-Bold
/Courier-BoldOblique
/Courier-Oblique
/Create
/CreationDate
/Creator
/CreatorInfo
/CropBox
/Cross
/Crypt
/CryptFilter
/CryptFilterDecodeParms
/Cyan
/D
/DA
/DCTDecode
/DL
/DTC
/DW
/DW2
/DamagedRowsBeforeError
/Darken
/Dashed
/Data
/Date
/Decimal
/Decode
/DecodeParams
/DecodeParms
/Default
/DefaultForPrinting
/Delete
/Departmental
/Desc
/DescendantFonts
/Descent
/Design
/Dest
/DestOutputProfile
/Dests
/DevDepGS_BG
/DevDepGS_FL
/DevDepGS_HT
/DevDepGS_OP
/DevDepGS_TR
/DevDepGS_UCR
/DeveloperExtensions
/DeviceCMY
/DeviceCMYK
/DeviceColorant
/DeviceGray
/DeviceN
/DeviceRGB
/DeviceRGBK
/Diamond
/Difference
/Differences
/DigestLocation
/DigestMethod
/DigestValue
/Dingbats
/DingbatsRot
/Direction
/Disc
/DisplayDocTitle
/Distribute
/Div
/DocMDP
/DocOpen
/Document
/Domain
/DotGain
/Dotted
/Double
/DoubleDot
/Draft
/Duplex
/DuplexFlipLongEdge
/DuplexFlipShortEdge
/E
/EF
/EFF
```

```
/EFOpen
/ETen-B5-H
/ETen-B5-V
/ETenms-B5-H
/ETenms-B5-V
/EUC-H
/EUC-V
/EarlyChange
/Ellipse
/EllipseA
/EllipseB
/EllipseC
/Encode
/EncodedByteAlign
/Encoding
/Encrypt
/EncryptMetadata
/End
/EndIndent
/EndOfBlock
/EndOfLine
/EntcryptMetaData
/Entrust.PPKEF
/ExData
/Exclude
/Exclusion
/Experimental
/Expired
/Export
/ExportState
/Ext-RKSJ-H
/Ext-RKSJ-V
/ExtGState
/Extend
/Extends
/ExtensionLevel
/Extensions
/ExternalOPIdicts
/ExternalRefXobjects
/ExternalStreams
/F
/F9+0
/FD
/FG
/FL
/False
/Ff
/FieldMDP
/Fields
/FillIn
/Filter
/Final
/First
/FirstChar
/FirstPage
/Fit
/FitB
/FitBH
/FitBV
/FitH
/FitR
/FitV
/FitWindow
/FixedPrint
/Fl
/Flags
/FlatDecode
/FlateDecode
/Font
/FontBBox
/FontDescriptor
/FontFamily
/FontFauxing
/FontFile
/FontFile2
/FontFile3
/FontMatrix
/FontName
/FontStretch
/FontWeight
/Footer
/ForComment
/ForPublicRelease
/Form
/FormEx
/FormType
/FreeText
/Frequency
/FullSave
/FullScreen
/Function
/FunctionType
/Functions
/G
/GBK-EUC-H
/GBK-EUC-V
/GBK2K-H
/GBK2K-V
/GBKp-EUC-H
/GBpc-EUC-H
/GBpc-EUC-V
/GTS_PDFA1
```

```
/GTS_PDFX
/Gamma
/Generic
/GenericRot
/GlyphOrientationVertical
/GoTo
/GoToRemoveActions
/Gray
/Green
/Groove
/Group
/H
/H1
/H2
/H3
/H4
/H5
/H6
/HF
/HKana
/HKanaRot
/HKscs-B5-H
/HKscs-B5-V
/HRoman
/HRomanRot
/HT
/Halftone
/HalftoneName
/HalftoneType
/Hanzi
/HardLight
/Header
/Headers
/Height
/Height2
/Help
/Helvetica
/Helvetica-Bold
/Helvetica-BoldOblique
/Helvetica-Oblique
/Hidden
/HideAnnotationActions
/HideMenubar
/HideToolbar
/HideWindowsUI
/Highlight
/HojoKanji
/Hue
/I
/IC
/ICCBased
/ID
/IDS
/IDTree
/IF
/IRT
/IT
/IX
/Identify
/Identify-H
/Identify-V
/Image
/ImageB
/ImageC
/ImageI
/ImageMask
/Import
/Include
/Ind
/Index
/Indexed
/Info
/Ink
/InkList
/Inline
/InlineAlign
/Insert
/Inset
/Intent
/InterPolate
/Interpolate
/InvertedDouble
/InvertedDoubleDot
/InvertedEllipseA
/InvertedEllipseC
/InvertedSimpleDot
/Invisible
/Issuer
/ItalicAngle
/JBIG2Decode
/JBIG2Globals
/JPXDecode
/JavaScriptActions
/Justify
/K
/KSC-EUC-H
/KSC-EUC-V
/KSCms-UHC-H
/KSCms-UHC-HW-H
/KSCms-UHC-HW-V
/KSCms-UHC-V
/KSCpc-EUC-H
```

/Kana
/Kanji
/Key
/KeyUsage
/Keywords
/Kids
/L
/L2R
/LBody
/LC
/LE
/LI
/LJ
/LL
/LLE
/LLO
/LW
/LWZDecode
/Lab
/Lang
/Language
/Last
/LastChar
/LastModified
/LastPage
/LaunchActions
/Layout
/Lbl
/Leading
/Legal
/LegalAttestation
/Length
/Length1
/Length2
/Length3
/Level1
/Lighten
/Limits
/Line
/LineHeight
/LineThrough
/LineX
/LineY
/Linearized
/ListMode
/ListNumbering
/Location
/Lock
/Locked
/LockedContent
/LowerAlpha
/LowerRoman
/LrTb
/Luminosity
/M
/MCID
/MCR
/MDP
/MK
/ML
/MMType1
/MN
/MacExpertEncoding
/MacRomanEncoding
/Magenta
/MarkInfo
/MarkStyle
/Marked
/Mask
/Matrix
/Matte
/MaxWidth
/Maxtrix
/Measure
/MediaBox
/Metadata
/Middle
/MissingWidth
/MixingHints
/ModDate
/Modify
/MovieActions
/Msg
/Multiply
/N
/NChannel
/NM
/Name
/Named
/Names
/NeedsRendering
/NewParagraph
/Next
/NextPage
/NoRotate
/NoView
/NoZoom
/NonEFontNoWarn
/NonEmbeddedFonts
/NonFullScreenPageMode
/NonStruct
/None

```
/Normal
/NotApproced
/NotForPublicRelease
/Note
/NumCopies
/NumberFormat
/Nums
/O
/OBJR
/OC
/OCG
/OCGs
/OCMD
/OCProperties
/OFF
/OID
/ON
/OP
/OPI
/OPM
/OS
/Obj
/ObjStm
/OneColumn
/Online
/Open
/OpenType
/OptionalContent
/Order
/Ordering
/Org
/Outlines
/OutputCondition
/OutputConditionIdentifier
/OutputIntent
/OutputIntents
/Outset
/Overlay
/Overline
/P
/PCM
/PDF
/PS
/PZ
/Padding
/Page
/PageElement
/PageLabel
/PageLabels
/PageLayout
/PageMode
/Pages
/Pagination
/PaintType
/Paragraph
/Parent
/ParentTree
/ParentTreeNextKey
/Part
/Pattern
/PatternType
/Perceptual
/Perms
/Pg
/PickTrayByPDFSize
/PieceInfo
/Placement
/PolyLine
/PolyLineDimension
/Polygon
/PolygonCloud
/PolygonDimension
/Popup
/PreRelease
/Predictor
/Preferred
/PresSteps
/PreserveRB
/Prev
/PrevPage
/Preview
/Print
/PrintArea
/PrintClip
/PrintPageRange
/PrintScaling
/PrinterMark
/PrintersMarks
/PrintingOrder
/Private
/ProcSet
/Process
/Producer
/Prop_AuthTime
/Prop_AuthType
/Prop_Build
/Properties
/Proportional
/ProportionalRot
/PubSec
/Q
```

```
/QuadPoints
/Quote
/R
/R2L
/RBGroups
/RC
/RD
/REx
/RI
/RIPEMD160
/RL
/RT
/Range
/ReadOnly
/Reason
/Reasons
/Receipients
/Rect
/Red
/Redition
/Ref
/Reference
/Registry
/RegistryName
/Rejected
/RelativeColorimetric
/Rendition
/Renditions
/Requirements
/Resources
/Rhombold
/Ridge
/RlTb
/Role
/RoleMap
/Root
/Rotate
/Round
/Row
/RowSpan
/Rows
/Ruby
/RubyAlign
/RubyPosition
/RunLengthDecode
/S
/SA
/SE
/SHA1
/SHA256
/SHA384
/SHA512
/SM
/SMask
/SMaskInData
/SS
/SV
/SVCert
/Saturation
/Schema
/Scope
/Screen
/Sect
/Separation
/SeparationColorNames
/SeparationInfo
/SetOCGState
/Shading
/ShadingType
/Sig
/SigFieldLock
/SigQ
/SigRef
/Signature
/SimpleDot
/Simplex
/SinglePage
/Size
/SoftLight
/Sold
/Solid
/Solidities
/Sort
/SoundActions
/SpaceAfter
/SpaceBefore
/Span
/SpawnTemplate
/SpotFunction
/Square
/Squiggly
/St
/Stamp
/Standard
/Start
/StartIndent
/State
/StemH
/StemV
/Stm
/StmF
/StmOwn
```

```
/StrF
/StrikeOut
/StructElem
/StructParent
/StructParents
/StructTreeRoot
/Style
/SubFilter
/SubType
/Subj
/Subject
/SubjectDN
/SubmitStandalone
/Subtype
/Summary
/SummaryView
/Supplement
/Suspects
/Sy
/Symbol
/T
/TBody
/TBorderStyle
/TD
/TFoot
/TH
/THead
/TK
/TOC
/TOCI
/TP
/TPadding
/TR
/TR2
/Table
/Tabs
/TbRl
/TemplateInstantianted
/Templates
/Text
/TextAlign
/TextDecorationColor
/TextDecorationThickness
/TextDecorationType
/TextIndent
/Thread
/Threads
/Thumb
/TilingType
/TimeStamp
/Times-Bold
/Times-BoldItalic
/Times-Italic
/Times-Roman
/Title
/ToUnicode
/Toggle
/ToggleNoView
/Top
/TopSecret
/Trans
/TransferFunction
/TransformMethod
/TransformParams
/Transparency
/TrapNet
/TrapRegions
/TrapStyles
/Trapped
/Trapping
/TrimBox
/True
/TrueType
/TrueTypeFonts
/TrustedMode
/Ttl
/TwoColumnLeft
/TwoColumnRight
/TwoPageLeft
/TwoPageRight
/Type
/Type0
/Type1
/Type1C
/Type3
/U
/UCR
/UCR2
/UR
/UR3
/URIActions
/Unchanged
/Underline
/UniCNS-UCS2-H
/UniCNS-UCS2-V
/UniCNS-UTF16-H
/UniCNS-UTF16-V
/UniGB-UCS2-H
/UniGB-UCS2-V
/UniGB-UTF16-H
/UniGB-UTF16-V
/UniJIS-UCS2-H
```

/UniJIS-UCS2-HW-H
/UniJIS-UCS2-HW-V
/UniJIS-UCS2-V
/UniJIS-UTF16-H
/UniJIS-UTF16-V
/UniKS-UCS2-H
/UniKS-UCS2-V
/UniKS-UTF16-H
/UniKS-UTF16-V
/Unknown
/Unmarked
/UpperAlpha
/UpperRoman
/Usage
/UseAttachments
/UseCMap
/UseNone
/UseOC
/UseOutlines
/UseThumbs
/User
/UserProperties
/UserUnit
/V
/V2
/VE
/VP
/VeriSign.PPKVS
/Version
/Vertices
/VerticesPerRow
/View
/ViewArea
/ViewClip
/ViewState
/ViewerPreferences
/Viewport
/VisiblePages
/W
/W2
/WMode
/Warichu
/Watermark
/WhitePoint
/Width
/Width2
/Widths
/WinAnsiEncoding
/WritingMode
/X
/XFAResources
/XHeight
/XML
/XObject
/XRef
/XRefStm
/XStep
/XYZ
/Xsquare
/Y
/YStep
/Yellow
/Ysquare
/ZapfDingbats
/Zoom
/adbe.pkcs7.detached
/adbe.pkcs7.sha1
/adbe.x509.rsa_sha1
/ca
/cb
/checked
/max
/min
/neutral
/null
/off
/on
/op
/pb
/rb
/tv

Microsoft Office elementer:

/Workbook
/Textbox
/Endnote
/Worksheet
/Macrosheet
/Annotation
/Dialogsheet
/Chartsheet
/Diagram
/Footnote
/Chart
/Slide
/InlineShape
/Artifact
/Figure
/Formula
/Link