PDF Advanced Electronic Signatures Profile of the OASIS DSS Version 1.0

Working Draft 01

DD Month 2014

Technical Committee:

OASIS Digital Signature Services eXtended (DSS-X) TC

Chairs:

Juan Cruellas, cruellas@ac.upc.edu, UPC-DAC
Stefan Drees, stefan@drees.name, Individual

Editors:

Martin Bosslet, martin.bosslet@gmail.com, Individual

Additional artifacts:

(leave empty if no additional artifacts) This prose specification is one component of a Work Product that also includes:

- XML schemas (list file names or directory name)
- Other parts (list titles and/or file names)

Related work:

(leave empty if no superseded or related works) This specification replaces or supersedes:

- specifications replaced by this specification (list, hyperlink if available)

This specification is related to:

- related specifications (list, hyperlink if available)

Declared XML namespaces:

- http://docs.oasis-open.org/dss-x/ns/pades

Abstract:

Summary of the technical purpose of the specification

Status:

This Working Draft (WD) has been produced by one or more TC Members; it has not yet been voted on by the TC or approved as a Committee Draft (Committee Specification Draft or a Committee Note Draft). The OASIS document Approval Process begins officially with a TC vote to approve a WD as a Committee Draft. A TC may approve a Working Draft, revise it, and re-approve it any number of times as a Committee Draft.

URI patterns:

Initial publication URI:
http://docs.oasis-open.org/dss-x/pades/v1.0/csd01/pades-v1.0-csd01.html
Permanent “Latest version” URI:
http://docs.oasis-open.org/dss-x/pades/v1.0/pades-v1.0.html
(Managed by OASIS TC Administration; please don’t modify.)

Table of Contents

1 Introduction

1.1 Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

1.2 Normative References

[RFC2119] Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels”, BCP 14, RFC 2119, March 1997. http://www.ietf.org/rfc/rfc2119.txt.

[Reference] [Full reference citation]

1.3 Non-Normative References

[Reference] [Full reference citation]

NOTE: The proper format for citation of technical work produced by an OASIS TC (whether Standards Track or Non-Standards Track) is:

[Citation Label]

Work Product title (italicized). Edited by Albert Alston, Bob Ballston, and Calvin Carlson. Approval date (DD Month YYYY). OASIS Stage Identifier and Revision Number (*e.g.*, OASIS Committee Specification Draft 01). Principal URI (version-specific URI, *e.g.*, with file name component: somespec-v1.0-csd01.html). Latest version: (latest version URI, without stage identifiers).

For example:

[OpenDocument-1.2]

Open Document Format for Office Applications (OpenDocument) Version 1.2. Edited by Patrick Durusau and Michael Brauer. 19 January 2011. OASIS Committee Specification Draft 07. http://docs.oasis-open.org/office/v1.2/csd07/OpenDocument-v1.2-csd07.html. Latest version: http://docs.oasis-open.org/office/v1.2/OpenDocument-v1.2.html.

2 Overview

3 PDF Advanced Electronic Signatures Profile

3.1 Overview

3.2 Profile Features

3.2.1 Scope

TODO: This profile explicitly covers PAdES signatures only. ISO 32000-1 type signatures are not covered. Only PAdES signatures in conformance with PAdES Part 3 (ETSI TS 102 778-3), PAdES Part 4 (ETSI TS 102 778-4) and the PAdES Baseline Profile (ETSI TS 103 172) are supported.

3.2.2 Relationship to other Profiles

TODO: Apart from the DSS Core Profile (http://docs.oasis-open.org/dss/v1.0/oasis-dss-core-spec-v1.0-os.html), this profile references elements from:

- the AdES Profile (http://docs.oasis-open.org/dss/v1.0/oasis-dss-profiles-AdES-spec-v1.0-os.html),
- the Signature Policy Profile (http://docs.oasis-open.org/dss-x/profiles/sigpolicy/oasis-dssx-1.0-profiles-sigpolicy-cd01.html) and
- the Visible Signature Profile (http://docs.oasis-open.org/dss-x/profiles/visualsig/v1.0/cs01/oasis-dssx-1.0-profiles-visualsig-cs1.html).

3.3 Profile of Signing Protocol

3.3.1 Attribute Profile

TODO: Define profile identifier

3.3.2 Element <SignRequest>

3.3.2.1 Element <InputDocuments>

TODO: PDFs must be provided as <Document> elements containing a <Base64Data> data holding the PDF in Base64-encoded form. The “MimeType” attribute, if set, must be set to “application/pdf”. If omitted, the server will assume that all <Base64Data> elements contain a PDF when using this profile. PDFs may also be provided as <AttachmentReference> elements.

TBD: Do we want to address the usage of <DocumentHash> for privacy concerns where the PDF cannot be sent to the server? As this use case cannot involve any specific PDF signature features (there's no document to add a visible signature field for instance), briefly mentioning the possibility should suffice?

Comment EJ: I would expect that the basic dss profile would suffice if only a DocumentHash is provided (for a PadES signature, the service will need the document).

3.3.2.2 Element <OptionalInputs>

3.3.2.2.1 New Optional Inputs

3.3.2.2.1.1 Optional Input <SuggestedDigestAlgorithm>

Clients may want to be able to choose a digest algorithm to be used for signature creation. However, it is not always possible to fully comply to such requests, e.g. the signature creation service might not be able to control the digest algorithms used by a timestamping service or it might simply not support the requested algorithm. The `<SuggestedDigestAlgorithm>` optional input MAY be supported by servers that choose to do so, the element serves as a hint only. Clients SHOULD NOT rely on support unless explicitly confirmed by server documentation or other out-of-band means of communication.

```
<xs:element name="SuggestedDigestAlgorithm”>
  <xs:complexType>
    <xs:complexContent>
      <xs:element ref=”ds:DigestMethod”/>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

Comment EJ: would it be better to just use the name DigestAlgorithm and that the service policy will indicate which digest algorithms are available (and which one is 'default'; these things have to be published by the service anyhow)? If the service cannot (or will not) use it, the element is ignored (it is optional anyhow).

Comment EJ: if the service will use it, how does the client known that it has been used? Does the client have to know? I would think so, otherwise it would rely on the service profile for the applied digest algorithm. (A verification could be done instead, but that seems a bit clumsy)

3.3.2.2.1.2 Optional Input <SignatureQualityLevel>

Some signature frameworks (cf. STORK QAA levels [TODO: link]) distinguish signatures by their level of quality, where a higher level of quality usually implies restrictions on how a signature is to be created. A server MAY offer clients to choose from different signature quality levels by implementing an application-dependent version of the generic `<SignatureQualityLevel>` optional input.

```
<xs:element name="SignatureQualityLevel”>
  <xs:complexType>
    <xs:sequence>
      <xs:element name=”Identifier” type=”xs:anyURI”/>
      <xs:element name=”Value” type=”dss:AnyType” minOccurs=”0”/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Comment EJ: What does this standardise? Because we have to consult the policy document of the service to find out which actual 'values' have to be used for the identifier & value elements. The likelyhood that two different services (from different providers) will use the same set of 'values' is low.. because they are not standardised. How to do conformance testing ? The phrase 'by implementing an application-dependent version of ...' should be left out..

3.3.2.2.2 Optional Inputs already defined

3.3.2.2.2.1 Optional Input <SignatureType>

TODO: Introduce new identifer for signature type “PAdES signature”. Optional, may be omitted as using this profile clearly indicates we're dealing with PAdES signatures.

Comment EJ: could there be different versions of PadES? Than a kind of version indicator can be useful. But otherwise it should not be needed.

3.3.2.2.2.2 Element <SignatureForm>

TODO: Introduce new signature form identifiers specifically related to PAdES signatures, including the forms introduced in the PAdES Baseline Profile (ETSI TS 103 172), and PadES-BES , PadES-EPES, PadES-LTV .. .

3.3.2.2.2.3 Element <SignedProperties>

TODO: Possible properties to support:

- Requesting SigningTime
- Requesting CommitmentTypeIndication (only for EPES-level signatures per PAdES specs)
- Requesting SignatureProductionPlace
- Requesting AllDataObjectsTimestamp (for content-timestamp)

TBD: Apart from the content-timestamp attribute, all the other properties are also available for inclusion in the visible signature profile. But since requesting inclusion of a content-timestamp attribute is a valid request, we might have to support <SignedProperties> anyway. I would tend towards including all of the above as a convenience feature for implementations that may already support the AdES profile.

If we agree on supporting them: Ambiguous requests where equivalent <SignedProperties> and Visible Signature Profile elements are included at the same time shall be rejected with a <Result> indicating an error. For example including a <RequestedSignatureProductionPlace> but also a Visible Signature Profile <Item> with <ItemName> equal to “SignatureProductionPlace” could lead to confusion and should thus be rejected.

I would tend to be as lenient as possible when supporting <SignedProperties>: Both the CAdES binary representation of the property to be included but also the XAdES XML representation should be allowed. If it were just the CAdES binary representation, clients would need to be able to generate valid ASN.1 data structures. Since it is possible to convert the XAdES representations of the properties to a valid CMS/CAdES representation, it might be more convenient for clients to specify properties on an XML basis instead of going through the ordeal of handling ASN.1/DER properly.

Comment EJ: is the use of CAdES binary representation or XAdES really needed?

3.3.2.2.2.4 Element <VisibleSignatureConfiguration>

3.3.2.2.2.4.1 Element <VisibleSignaturePolicy>

Update: It's not clear what it does when present in a request and may interfere with the signature-policy-identifier and commitment-type-indication signed properties. After discussing the element, we found it would be best to completely omit it. Any policy specifics should be handled by explicit signature-policy-identifier and commitment-type-indication

3.3.2.2.2.4.2 Element <FieldName>

TODO: Use as is. If absent, the server must be able to generate a unique name itself.

3.3.2.2.2.4.3 Element <VisibleSignaturePosition>

TBD: Neither <PixelVisibleSignaturePositionType> nor <GeneralVisibleSignaturePositionType> seem optimal. If forced to use them, I would opt for <GeneralVisibleSignaturePositionType> with all “measures” specified in “pt”, as this seems to be the most natural unit for PDF documents. To fully support what PDF offers in terms of positioning (cf. http://wwwimages.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/PDF32000_2008.pdf), we might even consider introducing rotations etc.

To keep things reasonable I would like to suggest a special <PdfVisibleSignaturePositionType> that is basically a GeneralVisibleSignaturePositionType without the need to specify “pt” as the unit measure (concrete coordinates could be specified as integer/decimal values)

Comment EJ: what about the page number where it has to be put? Or things like “last page”? How to standardise all the different possibilities? How does it relate to the 'Visible Signature Profile'?

3.3.2.2.2.4.4 Element <VisibleSignatureItemsConfiguration>

TBD: Do not support the <Item> with <ItemName> equal to “SignatureValue” as it is impossible for PDF signatures.

To prevent the server from having to be able to parse a multitude of date/time formats, I would like to constrain <ItemValueDate> elements to use xsd:dateTime format which should be the most widely supported.

Regarding <ItemPosition>, I would suggest to just allow <PercentItemPositionType> and a dedicated <PdfItemPositionType> following the thoughts outlined above w.r.t. a <PdfVisibleSignaturePositionType>.

Food for thought: Since we're targeting PDF directly, it could also be discussed to simplify the <VisibleSignatureItemsConfiguration> because the current generic format leads to quite deeply nested and verbose XML requests. If we can decide on a fixed number of items to support, the element could also be structured like this:

Note EJ: it would also need “width” and “height” … and <Page>..</Page>

Comment EJ: The VisibleSignaturePosition and VisibleSignatureItemsConfigurations have some overlap, for instance both specify something for a 'position'? Could it be that VisibleSignaturePosition is actually a kind of VisibleSignatureType?

```
<VisibleSignatureItems>
  <IncludeAttribute IncludeCaption=”true” Orientation=”90”>
    <Name>Subject:CommonName</Name>
    <Position>
      <X>15.5</X>
     <Y>30</Y>
    </Position>
  </IncludeAttribute>
  <IncludeAttribute>
    <Name>Subject:Organization</Name>
 </IncludeAttribute>
```

```
<SigningTime>
  <Value>2014-10-05T12:00:00+02:00</Value>
</SigningTime>
<Reason>
  <Value>Pick one</Value>
</Reason>
<Location>
  <Value>Saarbrücken, Germany</Value>
  <Position>
    <X>100</X>
    <Y>200</Y>
  </Position>
</Location>
<Image MimeType=”image/png”>
  <Value>aBARG...z65==</Value>
</Image>
</VisibleSignatureItems>
```

Advantages: The “IncludeCaption” and “Orientation” attributes are currently defined only globally, this way they would be defined on a per-item basis for the attributes to include. Parsing and locating individual items is easier, since there are dedicated elements for them with clearly defined formats. The XML structure is flattened, the current form leads to rather deeply nested elements. There's a disambiguating “MimeType” attribute for images. Consistent use of uppercase names for elements and attributes (as in the Core Profile).

Update: Further useful features:

- Besides free positioning of the elements and fixed “Orientation” values, free-form rotation could also be supported. We could support this in the <Position> element:

  ```
  <Position>
    <X>100</X>
    <Y>200</Y>
    <Rotation>45</Rotation>
  </Position>
  ```

- For included attributes, we should support choosing a prefix. E.g.

  ```
  <IncludeAttribute>
    <Name>Subject:Organization</Name>
  </IncludeAttribute>
  ```

  would display the certificate subject's organization in the visible signature as is.

<IncludeAttribute>

<Name>Subject:Organization</Name>

<IncludeCaption />

</IncludeAttribute>

An empty <IncludeCaption> element would display the organization with a default prefix, e.g. “Organization: <organization>”

<IncludeAttribute>

<Name>Subject:Organization</Name>

<IncludeCaption>Company:</IncludeCaption>

</IncludeAttribute>

This would display “Company: <organization>”

–

3.3.2.2.2.4.5 Element <other>

TBD: Should be named <Other> (uppercased) and should not be mandatory.

Comment EJ: Where is it used for?

3.3.2.2.2.5 Element <GenerateUnderSignaturePolicy>

TBD: May be used as is, although it would be nice for consistency reasons to have this as a <SignedProperties> element with similar syntax and semantics, i.e. requesting a signature policy would become a <Property> element in the <SignedProperties> optional input, allowing for both a CAdES binary or a XAdES XML representation of the requested signature policy.

3.3.2.2.3 Optional Inputs that should not be used

- `<Schemas>` (not applicable in PDF context)
- `<AddTimestamp>` (requesting timestamps is handled by `<SignatureForm>` and `<RequestAllDataObjectsTimestamp>`)
- `<IncludeObject>` (not applicable PDF context)
- `<IncludeEContent>` (forbidden by PAdES specs)
- `<SignaturePlacement>` (not applicable PDF context)
- `<SignedReferences>` (not applicable PDF context)
- `<Requesting DataObjectFormat>` (not applicable PDF context)

3.3.2.2.4 Optional Inputs that may be used as specified

- `<ServicePolicy>`
- `<ClaimedIdentity>`
- `<Language>`
- `<AdditionalProfile>`

- <IntendedAudience>
- <KeySelector>
- <Properties> (with respect to any restrictions imposed by PAdES specification)
- <ReturnSupportedSignaturePolicies>

3.3.3 Element <SignResponse>

3.3.3.1 Element <SignatureObject>

The <SignatureObject> SHOULD contain a <SignaturePtr> pointing to a <DocumentWithSignature> optional output that contains the signed PDF document.

Comment EJ: do we really need this? The core specifies a SignaturePtr as “This is used to point to an XML signature in an (..) output (for a sign response) document in which a signature is enveloped“ . I think it does not apply to the PAdES case, unless we want to be able to point to the actual signature INSIDE the pdf, but that is difficult (it would need the fieldname?)..

3.3.3.2 Element <OptionalOutputs>

3.3.3.2.1 Optional Outputs already defined

3.3.3.2.1.1 Element <DocumentWithSignature>

Always present if signature creation succeeded. Contains the signed PDF in a <Document> with <Base64Data> and optionally a MimeType attribute of “application/pdf”.

3.3.3.2.2 Optional Outputs that should not be used

- <Schemas> (not applicable PDF context)

3.3.3.2.3 Optional Outputs that may be used as specified

- <UsedSignaturePolicy>
- <SupportedSignaturePolicies>
- UsedDigestAlgorithm (see remark in Section 3.3.2.2.1.1)

3.4 Profile of Verifying Protocol

3.4.1 Attribute Profile

TODO: Use same identifier as for signing protocol.

3.4.2 Element <VerifyRequest>

3.4.2.1 Element <InputDocuments>

TODO: Two possible cases: Either there is a <SignatureObject> with a <SignaturePtr> element pointing to the <Document> under <InputDocuments> which contains the signed PDF file or the <SignatureObject> is omitted altogether with just an <InputDocuments> element containing a <Document> with the signed PDF file.

Comment EJ: I would expect only an InputDocuments element, because the other elements should contain/refer-to the actual signature (not just a 'document').

PDFs must be provided as <Document> elements containing a <Base64Data> data holding the PDF in Base64-encoded form. The “MimeType” attribute, if set, must be set to “application/pdf”. If omitted, the server will assume that all <Base64Data> elements contain a PDF when using this profile. PDFs may also be provided as <AttachmentReference> elements.

3.4.2.2 Element <SignatureObject>

See above.

3.4.2.3 Element <OptionalInputs>

3.4.2.3.1 Optional Inputs already defined

3.4.2.3.1.1 Element <ReturnUpdatedSignature>

TODO: Follows what is stated in the AdES Profile: Must be present when a client requests extension of a signature to one of the levels defined by the <SignatureForm> optional input. The “Type” attribute must match one of the <SignatureForm> identifiers.

Comment EJ: the SignatureForm element is not yet mentioned in the OptionalInputs for the VerifyRequest.

3.4.2.3.1.2 Element <FieldName>

TODO: Used as is, specify the fields to be verified.

TBD: What happens in the multi-verification case if multiple documents contain a field of the same name, but not all of them should be verified?

Comment EJ: than a reference to the actual document is needed as well... :-) or the service returns an error due to the ambiguity of the fieldname... But I think this is not reasonable: if we allow the verification of multiple documents in one verify request, we should also allow for the unambiguous specification of the fieldname. So the element fieldname should contain an optional idref to the document.

3.4.2.3.1.3 Element <VisibleIndicationFormat>

TBD: Probably not applicable or misleading?

3.4.2.3.2 Optional Inputs that should not be used

- `<VerifyManifests>` (not applicable PDF context)
- `<ReturnTransformedDocument>` (not applicable PDF context)
- `<ReturnTimestampedSignature>` (handled by signature extension using the `<SignatureForm>` optional input)

3.4.2.3.3 Optional Inputs that may be used as specified

- `<UseVerificationTime>`
- `<ReturnVerificationTimeInfo>`
- `<AdditionalKeyInfo>`
- `<ReturnProcessingDetails>`

- <ReturnSigningTimeInfo>
- <ReturnSignerIdentity>
- <VerifyUnderSignaturePolicy>

3.4.3 Element <VerifyResponse>

3.4.3.1 Element <OptionalOutputs>

3.4.3.1.1 Optional Outputs already defined

3.4.3.1.1.1 Element <UpdatedSignature>

TODO: If the original <VerifyRequest> included a <ReturnUpdatedSignature> optional input requesting signature extension, a successful response will include an <UpdatedSignature> optional output including a <SignatureObject> with a <SignaturePtr> pointing to a <DocumentWithSignature> containing the updated and extended PDF document as a <Document> with <Base64Data> and "MimeType" optionally set to "application/pdf". The "Type" attribute must match the identifier requested by the client in the <ReturnUpdateSignature> optional input.

3.4.3.1.1.2 Element <DocumentWithSignature>

See remarks for <UpdatedSignature>

3.4.3.1.1.3 Element <FieldName>

TODO: Must be present and indicates which signature fields have been verified.

Comment EJ: including a idref in case of multiple documents?

3.4.3.1.2 Optional Outputs that should not be used

- <VerifyManifestResults> (not applicable PDF context)
- <TransformedDocument> (not applicable PDF context)
- <TimestampedSignature>

3.4.3.1.3 Optional Outputs that may be used as specified

- <VerificationTimeInfo>
- <ProcessingDetails>
- <SigningTimeInfo>
- <SignerIdentity>
- <VerifiedUnderSignaturePolicy>

4 Mass Signing, Extension and Verification

TBD:

Update: After discussing this, we agreed that this should not be explicitly addressed in the PDF profile itself.

I'd like to address the use cases of being able to request signature creation, extension and verification of multiple PDF documents in a single request.

With the profile as discussed right now, mass signing could be implemented by just providing multiple PDFs under the <InputDocuments> element. Any <OptionalInputs> that are present would be assumed to apply globally for all signatures to be created on the provided input documents. Similarly, in the <SignResponse> elements, the <SignatureObject> could be omitted, all signatures created would be present in several <DocumentWithSignature> optional output elements.

Mass extension and verification are a bit more complicated as the Core Profile currently only tolerates a single <SignatureObject> in a <VerifyRequest>. As a workaround, it is typically possible to omit the <SignatureObject> altogether, and we could again provide several PDFs in the <InputDocuments> section. As in the signing case, any <OptionalInputs> would again apply on a global level. In the corresponding <VerifyResponse>, the extended documents could be made available as <DocumentWithSignature> optional outputs.

The trickiest part is probably error handling, as the single <Result> element is not sufficient to give detailed information about errors that occured for specific input documents. We could address this by defining new <OptionalOutputs> that allow addressing individual documents, possibly using a “WhichDocument” attribute as is often done in existing Profiles.

My question: Do we want to address these issues in this profile or should there be a separate “Mass Signing, Extension and Verification Profile”? Maybe that's the wiser approach, because it could be generalized to other signature types such as CAdES or XAdES signatures, too.

5 Identifiers defined in this Specification

6 Conformance

The last numbered section in the specification must be the Conformance section. Conformance Statements/Clauses go here.

Appendix A Acknowledgments

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Participants:

[Participant Name, Affiliation | Individual Member]
[Participant Name, Affiliation | Individual Member]

Appendix B Example Requests / Responses

Demonstrating appendix numbering.

Appendix B.1 Subsidiary appendix

text

Appendix B.1.1 Sub-subsidiary appendix

text

Appendix C Revision History

Revision	Date	Editor	Changes Made
[Rev number]	[Rev Date]	[Modified By]	[Summary of Changes]