DigiDoc Format Specification

Document version: 1.3.0, 12.05.2004

Newest specified format version: 1.3

This document describes the document format that is used in the DigiDoc system (hereinafter DIGIDOC-XML). DigiDoc uses XML as its base format and is based on the international standards XML-DSIG and ETSI TS 101 903.

References

RFC2560	Myers, M., Ankney, R., Malpani, A., Galperin, S., Adams, C., X.509 Internet Public Key Infrastructure: Online Certificate Status Protocol - OCSP. June 1999.
RFC3275	Eastlake 3rd D., Reagle J., Solo D., (Extensible Markup Language) XML-Signature Syntax and Processing. (XML-DSIG) March 2002.
ETSI TS 101 903	XML Advanced Electronic Signatures (XAdES). February 2002.
XML Schema 2	XML Schema Part 2: Datatypes. W3C Recommendation 02 May 2001 http://www.w3.org/TR/xmlschema-2/

Introduction

XML Advanced Electronic Signatures (XAdES) [ETSI TS 101 903] defines a format that enables structurally store signed data, signature and security attributes associated with digital signature (e.g. validity confirmations).

This document describes DigiDoc document format that is based on the XAdES standard and is a profile of that standard. The profile does not exactly match any subsets described in XAdES standard – the best format name would be “XAdES-C-L” indicating, that all certificates and OCSP confirmations are present but there are no “pure” timestamps. DigiDoc framework lays on PKI model where besides validity information, OCSP responses are treated as timestamps.

DigiDoc document format (DIGIDOC-XML) has the following important features:

- Can be verified offline without any additional information
- Signature can be given to several original documents at the same time
- Protection against format attacks – type of signed document is also signed
- Original document can be in the container or stored separately
- Original document can be XML or any binary file (Word, Excel, PDF, RTF etc)
- Zero, one or more signatures per container
- One validity confirmation per signature

Mandatory XAdES specification elements and attributes are used without deviations from the standard. A number of optional components from the XAdES specification are also used with the goal that long-time proof validity is ensured, and documents and signatures can be verified without any additional information. All the elements and attributes are specified below.

General DIGIDOC-XML document structure

DIGIDOC-XML file format structure is the following. We use notation defined in [RFC3275] chapter 2.

```
<?xml version="1.0" encoding="UTF-8" ?>
<SignedDoc format="DIGIDOC-XML" version="1.3"
     xmlns="http://www.sk.ee/DigiDoc/v1.3.0#">
  <!-- original data files  -->
  <DataFile />
  <!-- client signatures with validity confirmations -->
  <Signature />
</SignedDoc>
```

Therefore, a DIGIDOC-XML file is basically a <SignedDoc /> container that contains original data files and signatures.

Original data – One or more original data files or references to external files.

Signatures – One or more signatures that confirm the integrity of all the data files or external referenced files contained in the document. If the document contains more than one data file or reference to external file, each signature must confirm the checksums and data types of all the files. The signature also contains validity confirmation. One validity confirmation confirms the validity of one specific client signature at given moment of time. All signatures must be accompanied by validity confirmations – no signatures without validity confirmations may be added to the document.

The full general structure of DIGIDOC-XML format is as follows.

```
<SignedDoc format= version= xlmns=>
 (<DataFile Id= Filename= ContentType= MimeType= Size= DigestType= DigestValue=
>)+
  <Signature Id= xmlns=>
   <SignedInfo>
    <CanonicalizationMethod Algorithm= >
    <SignatureMethod Algorithm= >
    (<Reference URI= >
     (<Transforms>
      <Transform Algorithm= >
     </Transforms>)?
     <DigestMethod Algorithm= >
     <DigestValue />
    </Reference>)+
   <SignedInfo xmlns=>
   <SignatureValue Id= >
```

```
<KeyInfo>
 <KeyValue>
   <RSAKeyValue>
     <Modulus />
     <Exponent />
   </RSAKeyValue>
 </KeyValue>
 <X509Data>
  <X509Certicate />
 </X509Data>
</KeyInfo>
<Object>
 <QualifyingProperties xmlns= Target= >
  <SignedProperties Id= >
   <SignedSignatureProperties>
    <SigningTime />
    <SigningCertificate>
     <Cert Id= >
      <CertDigest>
       <DigestMethod Algorithm= />
       <DigestValue />
      </CertDigest>
      <IssuerSerial>
       <X509IssuerName xmlns= />
       <X509SerialNumber xmlns= />
      </IssuerSerial>
     </Cert>
    </SigningCertificate>
    <SignaturePolicyIdentifier>
     <SignaturePolicyImplied/>
    </SignaturePolicyIdentifier>
    (<SignatureProductionPlace>
     <City />
     <StateOrProvince />
     <PostalCode />
     <CountryName />
    </SignatureProductionPlace>)?
    (<SignerRole>
     <ClaimedRoles>
      <ClaimedRole />
     </ClaimedRoles>
    </SignerRole>) ?
   </SignedSignatureProperties>
   <SignedDataObjectProperties />
  </SignedProperties>
  <UnsignedProperties>
   <UnsignedSignatureProperties>
    <CompleteCertificateRefs>
     <CertRefs>
      <Cert>
       <CertDigest>
        <DigestMethod Algorithm= />
        <DigestValue />
       </CertDigest>
       <IssuerSerial>
        <X509IssuerName xmlns= />
        <X509SerialNumber xmlns= />
       </IssuerSerial>
      </Cert>
     </CertRefs>
    </CompleteCertificateRefs>
    <CompleteRevocationRefs>
     <OCSPRefs>
```

```
        <OCSPRef>
         <OCSPIdentifier URI= >
          <ResponderID />
          <ProducedAt />
         </OCSPIdentifier>
         <DigestAlgAndValue>
          <DigestMethod Algorithm= />
          <DigestValue />
         </DigestAlgAndValue>
        </OCSPRef>
       </OCSPRefs>
      </CompleteRevocationRefs>
      <CertificateValues>
       <EncapsulatedX509Certificate Id= />
      </CertificateValues>
      <RevocationValues>
       <OCSPValues>
        <EncapsulatedOCSPValue Id= >
       </OCSPValues>
      </RevocationValues>
     </UnsignedSignatureProperties>
    </UnsignedProperties>
   </QualifyingProperties>
  </Object>
 </Signature>
</SignedDoc>
```

Elements and their attributes

Root element (SignedDoc)

The root element of each DigiDoc file is **<SignedDoc>.** It has the following attributes:
format – DigiDoc file format name. Current format is "DIGIDOC-XML". The older format "SK-XML" is also known.
version – DigiDoc file format version. Current version is "1.3". In case of the older format "SK-XML", version is "1.0" or in case of DIGIDOC-SK format older versions are 1.1 and 1.2. The current document describes the format 1.3. See below for list of differences between versions.
xmlns – SignedDoc namespace is required http://www.sk.ee/DigiDoc/v1.3.0#

Original data files (DataFile)

A DigiDoc file contains one or more original data files or references to external files. For each file, a **<DataFile>** record is present. It has the following attributes:

- **Id** – unique file identifier within this document. Data file identifiers begin with the character 'D' followed by file sequence number.
- **Filename** – actual (external) file name without path.
- **ContentType** – document encapsulation method (DETATCHED, EMBEDDED_BASE64 or EMBEDDED)
    - **EMBEDDED** – original file data is in original format and is enclosed in this record. Can only be used in case of XML original data. The

original XML data may not contain XML header (<?xml … ?> or DTD. XML elements described within this document are allowed. One DigiDoc file can be embedded in another DigiDoc file in original format.

- **EMBEDDED_BASE64** – file data are enclosed in this record as Base64-encoded data.
- **DETATCHED** – original data is contained in the file whose name is specified in the Filename attribute.

- **MimeType** – data type of original data.
- **Size** – size of original data file in bytes.
- **DigestType** - original data file hash type. Currently, only SHA1 is supported. Only required in case of DETATCHED original file.
- **DigestValue** – original data file digest value encoded in Base64. Digest is calculated across original data in original format. Only required in case of DETATCHED original file.
- **xmlns -** SignedDoc namespace is required http://www.sk.ee/DigiDoc/v1.3.0#
- Arbitrary number of other attributes (metadata) in the format <name>="<value>".

Signatures (Signature)

A DIGIDOC-XML file may contain any number of signatures. Each signature is specified in a <Signature> block. Its main structure elements are:

- <SignedInfo> - XML-DSIG block that contains the info to be signed
- <SignatureValue> - actual signature
- <KeyInfo> - the certificate used to give the signature and its RSA public key.
- <Object> + <QualifyingProperties> - extension block according to XAdES. It contains:
  - <SignedProperties>+<SignedSignatureProperties> - additional data to be included in the signature. They are:
    - signing time (<SigningTime>)
    - info about certificate used to give the signature (<SigningCertificate>)
    - signature policy (<SignaturePolicyIndentifier>)
    - place of signing (<SignatureProductionPlace>)
    - signer role (<SignerRole>)
  - <UnsignedProperties>+<UnsignedSignatureProperties> - unsigned data. They are:
    - validity confirmation server (OCSP server) certificate info (<CompleteCertificateRefs>)
    - validity confirmation info (<CompleteRevocationRefs>+<OCSPRefs>)
    - validity confirmation server (OCSP responder) certificate (<CertificateValues>)
    - actual validity confirmation (<RevocationValues>)

Signature parameters

Client signature is contained in the <Signature> element. This element has the following attributes:

- **Id** – unique client signature identifier in this document. Client signature identifies begin with the 'S' character, followed by signature sequence number.
- **xmlns** – XML signature namespace. Must have this value: "http://www.w3.org/2000/09/xmldsig#".

Signed information block (SignedInfo)

All data signed by the client are contained in the <SignedInfo> block. The <SignedInfo> element may have an attribute "xmlns" with the same content as <Signature> element. If it is not present, the library automatically adds it for hash calculation. DigiDoc file signatures are canonized and the signature method is always SHA1 + RSA. This is reflected by the elements <CanonicalizationMethod> and <SignatureMethod> in the beginning of this block. They have the following fixed content:

```
    <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
```

One <Reference> block for each data file follows. For each signature, another <Reference> block is present, to store the hash code of the info to be signed, as specified by XAdES extensions. The <Reference> block concerning the original data file contains the hash code of <DataFile> and the hash code of the signed info specified by XAdES extensions and present in <SignedProperties> block. Each <Reference> element has an URI attribute. The value of it is as follows, according to block type:

- URI="#<doc-id>" – Hash code of original data.
- URI="#<signature-id>-SignedProperties" – Hash code of additional info to be added to signature according to XAdES extensions. The element <Reference> that contains the hash code of element <SignedProperties> must also have a "Type" attribute with the following value: "http://uri.etsi.org/01903/v1.1.1#SignedProperties".

Hash code type is always sha1 and signature type is detatched-signature, i.e. the <Reference> elements contain the hash codes of <DataFile> elements contained in this document. This is specified by <Reference> element attribute URI="<xml-block-id>".

The element <Transforms> is only necessary if the <Reference> block contains hash code of a data file with embedding type DETATCHED. In this case, the element <DataFile> has only a reference type to the external file and its digest code in the attribute Digest Value. In each <Reference> block, there is also a <DigestValue> element that contains the hash code of <DataFile> element in Base64. In this case, <Reference> child elements <DigestMethod> and <Transforms> are required. They must have the following values:

```
      <Transforms>
        <Transform
Algorithm="http://www.sk.ee/2002/10/digidoc#detatched-document-
signature"/>
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
```

Actual signature (SignatureValue)

For creating a signature, signed data hash codes must be collected and stored in the <SignedInfo> block in the format describe above. From this element, a signature is created and stored in <SignatureValue> element as Base64 encoded data. This element has an attribute "id" with this format: id="<signature-id>-SIG".

Signer certificate (KeyInfo)

At the end of client signature, signer certificate in Base64 (PEM format) is present in two representations:

- <KeyValue>/<RSAKeyValue> contain the identifying subelements <Modulus> and <Exponent>
- <X509Data>/<X509Certificate> contain the actual certificate in Base64 format.

XAdES extension block – parameters to be signed (SignedSignatureProperties)

The signature is followed by signature properties that are encapsulated in <Object> and <QualifyingProperties> elements. Parameters of <QualifyingProperties> elements are:

- **Target** – value pointing to signature in the format "#<signature-id>"
- **xmlns** – XML namespace. Value http://uri.etsi.org/01903/v1.1.1# is required.

There are two types of properties – signed and unsigned.

Signed properties are stored in the <SignedProperties> element and are protected against changes with a hash code that is signed by client and has the identifier URI="#<signature-id>-SignedProperties". <SignedProperties> parameter is:

- **Id** – identifier matching the signature in the format "<signature-id>-SignedProperties"

Current format supports only those signed signature properties that are stored in the <SignedSignatureProperties> element:

- Signer computer time at time of signing - <SigningTime>. Stored in dateTime format (see XML Schema2 p.3.2.7): "YYYY-MM-DDTHH24:MM:SS(+/-)TZ:00".
  Starting from version DIGIDOC-XML 1.1, all times are stored as UTC and time zone value is 'Z'.

- Client certificate info – to protect against changing client certificate, we sign the following attributes and store them in the element <Cert Id="<signature-id>-CERTINFO">:
    - certificate digest code - <CertDigest>/<DigestValue> - always SHA1 digest.
    - DN of certificate issuer - <IssuerSerial>/<X509IssuerName>
    - certificate identifier - <IssuerSerial>/<X509SerialNumber>
- Signature usage rules - <SignaturePolicyIdentifier>. This is mandatory according to XAdES. DIGIDOC-XML uses the value <SignaturePolicyImplied />. It means that signing policy is specified outside the document – in case of the Estonian ID card, by Digital Signature Law, SK Certificate Practice Statement and ESTEID Certificate Policy.
- Signing place as nonmandatory element - <SignatureProductionPlace>. The following data are stored here in arbitrary format:
    - Name of city - <City>
    - State or province - <StateOrProvince>
    - Postal code - <PostalCode>
    - Country name - <CountryName>
- Signer role as nonmandatory element – <SignerRole> - the roles that the signer himself claims to have (unverified). DIGIDOC-XML only uses the claimed role element <ClaimedRoles>. There may be one or more of them. DIGIDOC-XML also allows this field to be interpreted as resolution that is added at time of document signing.

XAdES extension block – unsigned parameters (UnsignedSignatureProperties)

The unsigned properties are stored in the <UnsignedProperties> element. The current version only supports unsigned signature properties <UnsignedSignatureProperties>. Here, references to OCSP responder certificate and validity confirmation and the actual certificate and validity confirmation are stored.

First, validity confirmation issuer certificate properties are stored in the element <CompleteCertificateRefs>. The element <Cert Id="<signature-id>-RESPONDER_CERTINFO"> stores the following certificate properties:

- certificate digest code - <CertDigest>/<DigestValue> - always SHA1 digest.
- DN of certificate issuer - <IssuerSerial>/<X509IssuerName>
- certificate identifier - <IssuerSerial>/<X509SerialNumber>

Following is the element <CompleteRevocationRefs>. It contains validity confirmation issuer, timestamp and the digest code in the subelement <OCSPRefs>/<OCSPRef>.

Validity confirmation issuer data are stored in the element <OCSPIdentifier> with the attribute:

- URI – reference to actual validity confirmation in the format #<validity-confirmation-id >

and subelements:

- ResponderID – OCSP server identifier, for example: "/C=EE/O=ESTEID/OU=OCSP/CN=ESTEID-SK OCSP RESPONDER/emailAddress=pki@sk.ee".
- ProducedAt – validity confirmation issuing time in dateTime format: "YYYY-MM-DDTHH24:MM:SS(+/-)TZ:00".

Validity confirmation digest code is stored in <OCSPRef> subelement <DigestAlgAndValue> with subelements:

- DigestMethod – specifies digest calculation algorithm with this attribute:

  Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"

- DigestValue – digest value in Base64.

The validity confirmation issuer certificate is stored as Base64 (PEM) in the element <CertificateValues> <EncapsulatedX509Certificate Id="<signature-id>-RESPONDER_CERT">.

Finally, validity confirmation data is stored in the element <RevocationValues>/<OCSValues>. This is followed by the element <EncapsulatedOCSPValue> with the attribute Id – validity confirmation identifier.

The content of the <EncapsulatedOCSPValue> element is OCSP validity confirmation in Base64 format. The OCSP request should contain hash of the Signature value in the Nonce field. The hash over Signature Value should be calculated over 128-byte value of the RSA signature (not over the base64 form). This Nonce value is also contained in the OCSP response.

Verification rules

The signed document should contain one or more original data files (<DataFile> element) and could contain one ore more signatures (<Signature> element). Every signature in the document should be given over all original files. Additionally, the signature is calculated over XAdES <SignedProperties> extensions block. For verification of DIGIDOC-XML document, the following steps should be performed:

1. **SignedDoc** – verify that the root element is <SignedDoc>
2. **Format and version** – check whether if the format and version are known
3. **Digests of original files** – calculate the hash digests of every original data file using canonized form of <DataFile> block.

4. **Detached files** – if any of <DataFile> element's parameter ContentType has value DETACHED then the original file is separate of container. The file should be located using parameter Filename, calculate the hash digest and compare it to Digest parameter. If the file cannot be located then user application should be able to feed the correct location.
5. **SignedInfo** – Every signature must contain element <SignedInfo> that contains one <Reference> element per original data file. In addition to checking proper syntax, hash values of <DigestValue> elements should be verified. One of them is referring to digest value of original data file; the other is referring to the hash value of <SignedProperties> element.
6. **Signature** - The hash value over canonized form of <SignedInfo> block is encrypted with signer's private key. Resulting 128-byte RSA-SHA1 signature is saved in <SignatureValue> element in base64 form. Every signature must contain certificate of the signer in element <X509Certificate>. The private component should be extracted from this certificate in order to decrypt the RSA signature. The result is 35-byte ASN.1-encoded value which contains signed hash value in 20 last bytes. This hash should match with hash calculated over <SignedInfo> block.
7. **Certificate digest** – The hash digest of the certificate stored in element <SigningCertificate>/<Cert>/<CertDigest>/<DigestValue> should match with hash of the <X509Certificate> content. Issuer of the signer's certificate should be recognized.
8. Validity confirmation – Every signature should have one validity confirmation (OCSP response) which is stored in <EncapsulatedOCSPValue>. The response field Nonce should match with hash of <SignatureValue> element content. The OCSP response is signed and corresponding certificate is available in element <EncapsulatedX509Certificate>.

Differences between DIGIDOC-XML versions 1.0 and 1.1

The document format version 1.1 provides better interoperability with existing XML-DSIG applications and fixes some minor errors from 1.0:

- The previous version had "SK-XML" as the format identifier and the only allowed version was "1.0". The newer format has "DIGIDOC-XML" as the format identifier, and currently the only allowed version is "1.1".
- In version 1.0, the DigestType and DigestValue attributes of <DataFile> element were mandatory. They contained the digest of the element contents (without XML tags). In version 1.1, these attributes are only required if the <DataFile> element points to an external file (attribute ContentType="DETATCHED") and they contain the digest of the external file specified in the Filename attribute.
- In version 1.0, the digest of the <DataFile> element was calculated in the original format. In version 1.1, the digest is calculated across the whole <DataFile> element in canonized form. Due to this, in version 1.1 there are no longer any

unsigned attributes of the <DataFile> attribute. In version 1.0, the signed attributes were MimeType and the digest of original data.

- In version 1.0, the signed info block <SignedInfo> contained two <Reference> blocks per each <DataFile> element. In version 1.1, <SignedInfo> contains only one <Reference> block per each <DataFile> element. The digest stored in there is calculated across the whole <DataFile> in canonized format. As the whole <DataFile> is signed in version 1.1, there is no need for separate <Reference> block with MIME type digest.
- In version 1.0, each <Reference> block also contained XML-DSIG enveloped-signature transformation record. This is not used in version 1.1. An exception to this is <DataFile> elements referencing to external files. (ContentType="DETATCHED"). In this case, the <Reference> block must contain the transformation with the algorithm: http://www.sk.ee/2002/10/digidoc#detatched-document-signature.
- In version 1.0, the element <X509Certificate> had an Id attribute. This is not used in version 1.1.
- In version 1.1, the modulus and exponent of public key in signer certificate are stored in element <KeyInfo>/<KeyValue>/<RSAKeyValue> subelements <Modulus> and <Exponent>.
- In version 1.0, the dateTime structure (used in <SigningTime> and <ProducedAt> elements) had the + and - signs mixed up in timezone notation. In version 1.1, time is always stored in UTC time zone.
- Syntax error in <SignedProperties> attribute „Id“ – had an extra # sign in the beginning (actually not necessary)
- Syntax error in <SigningCertificate> subelement <Cert> attribute "Id" – had an extra # sign in the beginning (actually not necessary)
- DIGIDOC-XML version 1.1 is fully compliant with the XML-DSIG standard, 1.0 was not. For testing, e.g. Apache XML Security package can be used. Problems will only arise with detached-document-signature transformation specific to external reference files; also, Apache does not check validity confirmations etc.

Differences between DIGIDOC-XML versions 1.1 and 1.2

- The previous format identifier was "DIGIDOC-XML" and the only allowed version was "1.1". The newer version identifier is "DIGIDOC-XML" and the allowed versions are "1.1" and "1.2".
- In the previous element, <SignedInfo> and <SignedProperties> elements required the attribute "xmlns" with the value http://www.w3.org/2000/09/xmldsig#. In version "1.2", this attribute is required only for <Signature> element. For all its subelements, it is allowed but not required. At the time of digest calculation, this element is automatically generated to main XML block element (<SignedInfo> and <SignedProperties>). This is compliant with the behavior of Apache XML Security

and .NET.

- In version "1.2", for the element <Reference> containing the digest of <SignedProperties>, the attribute "Type" is required with this value: http://uri.etsi.org/01903/v1.1.1#SignedProperties.
- In version "1.1", only the most important attributes of the OCSP validity confirmation were stored in memory, and not the whole confirmation. Due to this, an error could be generated at the time of file saving where validity confirmation value and its digest did not match. This is fixed in version "1.2".

Differences between DIGIDOC-XML versions 1.2 and 1.3

1.3 version of DIGIDOC-XML was created in order to fix some syntactical glitches in order to achieve full and proven interoperability with other XAdES implementations. The mistakes corrected are:

- <QualifyingProperties> - version 1.2 had no parameters. By XAdES syntakx parameters xmlns="http://uri.etsi.org/01903/v1.1.1#" ja Target="#<signature-id>" are required
- <SignedProperties>. Parameters xmlns= and Target= should be not allowed.
- <IssuerSerial>. In version 1.2 it contained serial number of the certificate. In 1.3 it contains:

```
<IssuerSerial>
  <X509IssuerName xmlns="http://www.w3.org/2000/09/xmldsig#">
    <!-- issuer DN of the  signer's certificate -->
  </X509IssuerName>
  <X509SerialNumber>
    <!-- serial number of the signer's certificate -->
  </X509SerialNumber>
</IssuerSerial>
```

- Syntax of dateTime data type is corrected: '-' should be used in date instead of '.'. This change affects elements <SigningTime> and <ProducedAt>.
- Parameter Id= is removed from element <Cert> as it is not allowed
- Parameter Target= is removed from element <UnsignedProperties>
- Element <CertRefs> is required between parameters <CompleteCertificateRefs> and <Cert>
- Element <OCSPValues> is required between parameters <RevocationValues> and <EncapsulatedOCSPvalue>
- All documents should be referring to SignedDoc namespace http://www.sk.ee/DigiDoc/v1.3.0#.

DigiDoc format definition in XML Schema

DigiDoc format is based on the standard ETSI TS 101 903 – XML Advanced Electronic Signatures (XAdES), that is an extension to the standard XML-DSIG – XML -Signature Syntax and Processing. DigiDoc is a profile of XAdES standard, presenting some extra requirements. Following is the full DigiDoc format specification in XML Schema, with details about DigiDoc and XAdES format differences.

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

<xsd:schema targetNamespace="http://www.sk.ee/DigiDoc/v1.3.0#"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       xmlns="http://www.sk.ee/DigiDoc/v1.3.0#"
       xmlns:etsi="http://uri.etsi.org/01903/v1.1.1#"
       xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
       elementFormDefault="qualified" >

<xsd:import namespace="http://www.w3.org/2000/09/xmldsig#"
schemaLocation="xmldsig-core-schema.xsd"/>

<!-- Root element for SignedDoc -->

<xsd:element name="SignedDoc" type="SignedDocType"/>
<xsd:complexType name="SignedDocType">
       <xsd:sequence>
              <xsd:element name="DataFile" type="DataFileType"
                     minOccurs="1" maxOccurs="unbounded"/>
              <xsd:element ref="ds:Signature"
                     minOccurs="0" maxOccurs="unbounded"/>
       </xsd:sequence>
       <xsd:attribute name="format" type="xsd:string" fixed="DIGIDOC-XML"/>
       <xsd:attribute name="version" type="xsd:string" fixed="1.3"/>
</xsd:complexType>

<!-- payload data - DataFile -->

<xsd:complexType name="DataFileType">
  <xsd:simpleContent>
       <xsd:extension base="xsd:string">
              <xsd:attribute name="Id" type="xsd:ID" use="required"/>
              <xsd:attribute name="Filename" type="xsd:string" use="required"/>
              <xsd:attribute name="ContentType">
              <xsd:simpleType>
```

```
                    <xsd:restriction base="xsd:string">
                          <xsd:enumeration value="EMBEDDED"/>
                          <xsd:enumeration value="EMBEDDED_BASE64"/>
                          <xsd:enumeration value="DETACHED"/>
                    </xsd:restriction>
              </xsd:simpleType>
              </xsd:attribute>
              <xsd:attribute name="MimeType" type="xsd:string" use="required"/>
              <xsd:attribute name="Size" type="xsd:decimal" use="required"/>
       <!-- but required for DETATCHED files -->
              <xsd:attribute name="DigestType" type="xsd:string" use="optional"/>
              <xsd:attribute name="DigestValue" type="xsd:string" use="optional"/>
       </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

</xsd:schema>
```