Electronic PostMark (EPM) Profile of the OASIS Digital Signature Service

Committee Draft, 24 December, 2004 (Working Draft 06)

Document identifier:

oasis-dss-1.0-profiles-epm-spec-cd-01

Location:

http://docs.oasis-open.org/dss/

Editor:

Ed Shallow, *Universal Postal Union* <ed.shallow@rogers.com>

Contributors:

Trevor Perrin, *individual* <trevp@trevp.net>
Nick Pope, individual <pope@secstan.com
Juan Carlos Cruellas, *individual* <cruellas@ac.upc.es>

Abstract:

This draft defines a profile of the OASIS DSS protocol for the purpose of creating and verifying signatures and timestamps which support the extended features of the Universal Postal Union's Electronic PostMarking service.

Status:

This is a **Working Draft** produced by the OASIS Digital Signature Service Technical Committee. Committee members should send comments on this draft to dss@lists.oasis-open.org.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Digital Signature Service TC web page at http://www.oasis-open.org/committees/dss/ipr.php.

Table of Contents

1 Introduction

The Electronic PostMarking service is a Universal Postal Union (UPU) endorsed standard aimed at providing generalized signature creation, signature verification, timestamping, and receipting services for use by and across Postal Administrations and their target customers.

Although the total scope and functional coverage of the EPM's service offering are outside the immediate scope of the DSS initiative, the UPU wishes to offer its client base a DSS-compliant subset of the EPM for clients who wish to maintain OASIS compliance in the core areas of signature and timestamp, creation and verification. This profile can be used directly as the basis for implementing interoperable systems.

1.1 Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD",

"SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as described in IETF RFC 2119 **[RFC 2119]**. These keywords are capitalized when used to unambiguously specify requirements over protocol features and behavior that affect the interoperability and security of implementations. When these words are not capitalized, they are meant in their natural-language sense.

This specification uses the following typographical conventions in text: `<ns:Element>`, `Attribute`, Datatype, `OtherCode`.

1.2 Namespaces

The structures described in this specification are contained in the schema file **[EPM]**. All schema listings in the current document are excerpts from that schema file. In the case of a disagreement between the schema file and this document, the schema file takes precedence.

This schema is associated with the following XML namespace:

```
http://www.docs.oasis-open.org/dss/oasis-dss-1.0-profiles-EPM-cd-01#
```

If a future version of this specification is needed, it will use a different namespace.

Conventional XML namespace prefixes are used in this document:

- The prefix `dss:` (or no prefix) stands for the DSS core namespace **[Core-XSD]**.
- The prefix `dsig:` stands for the W3C XML Signature namespace **[XMLSig]**.
- The prefix `xs:` stands for the W3C XML Schema namespace **[Schema1]**.
- The prefix `saml:` stands for the OASIS SAML Schema namespace **[SAMLCore1.1]**.
- The prefix `epm:` stands for the EPM Schema namespace **[EPM]**.
- The prefix xades: stands for ETSI XML Advanced Electronic Signatures (XAdES) document **[XAdES]**.

Applications MAY use different namespace prefixes, and MAY use whatever namespace defaulting/scoping conventions they desire, as long as they are compliant with the Namespaces in XML specification **[XML-ns]**.

2 Profile Features

2.1 Identifier

urn:oasis:names:tc:dss:1.0:profiles:epm

2.2 Scope

This document profiles the DSS signing and verifying protocols defined in **[DSSCore]** and provides an OASIS DSS-compliant interface to selected services of the EPM.

The EPM profile supports the creation and verification of both CMS/PKCS7 and **[XMLSig]** signature types.

Additional services within the EPM are supported through the extensibility mechanisms provided by the optional inputs and outputs of the **[DSSCore].** This includes:

- Easy to use EPM Signing Templates
- PostMarked receipts
- Embedding of signatures in documents
- Extended support of both XMLSig and CMS signature creation and verification
- Certificate validation data
  - Revocation references
  - Certificate references
  - Online Certificate Status Protocol (OCSP) responses
- Support for the chaining of complex business lifecycle events
- Timestamping from a CA-independent TimeStamp Authority

2.3 Relationship To Other Profiles

The profiles in this document are based directly on the **[DSSCore]**.

2.4 Signature Object

This profile supports the creation and verification of both XMLSig and CMS signatures as defined in **[DSSCore]**.

2.5 Transport Binding

This profile is transported using either an XML-based HTTP payload POSTed to the EPM Server, or via a SOAP Transport Binding as defined in the OASIS EPM Profile Web Service Description language (WSDL).

2.6 Security Binding

2.6.1 Security Requirements

The TLS X.509 Server Authentication security binding as described in section 6.2.1 in **[DSSCore]** must be used**.**

2.7 Common Elements

This section describes elements used and referenced within both the Sign and Verify protocols.

2.7.1 Element <PostMarkedReceipt> and the PostMarkedReceipt Signature

A PostMarkedReceipt is a signature attesting to the validity of either the signature just created (Sign protocol) or the signature just verified (Verify protocol). It requires an additional profile element not part of **[DSSCore]** and that is the <PostMarkedReceipt> element. This element describes the EPM's receipt structure, which works in conjunction with the standard <TstInfo> element of **[DSSCore].** A PostMarkedReceipt signature is returned whenever the optional input <IssuePostMarkedReceipt> is included in either the Sign or Verify request. The PostMarkedReceipt is a superset of the DSS <Timestamp> element and carries specific meaning within the specific context of EPM service provisioning. Semantics as follows:

- **Sign**

  When a PostMarkedReceipt signature is issued as a result of a Sign operation, the EPM is attesting to the origin of the signature and the validity of the certificate used to create it.

- **Verify**

  Correspondingly, when the EPM issues a PostMarkedReceipt as a result of a Verify operation which requested an <IssuePostMarkedReceipt>, the EPM is attesting to the validity of both the verified signature as well as the validity (i.e. revocation status) of the public verification certificate contained therein.

See section 6 for a detailed example of a standalone PostMarkedReceipt signature returned after successful verification. The example illustrates a detached receipt signature representing the PostMark covering a signed and verified document. Additionally, all evidence surrounding this event is logged in the EPM's non-repudiation database by default.

The EPM supports the issuance of conventional timestamps, both embedded and standalone. The EPM-specific notion of a PostMarked receipt applies in both the embedded and standalone scenarios. Both are valid within the Sign protocol.

All receipts are tied to a specific EPM operational transaction as specified by the enclosed <TransactionKey> element.

The <PostMarkedReceipt> element is similar to the <dss:Timestamp> when applied to XMLSig-based signatures.

PostMarkedReceipt signatures returned in XMLSig signatures scenarios, are exactly three (3) <dsig:Reference>'s which make up the signature associated with the PostMarkedReceipt. They are as follows:

- <dsig:Reference> whose URI attribute references a <dsig:Object> containing the <TstInfo>
- <dsig:Reference> whose URI attribute references a <dsig:Object> containing the <epm:PostMarkedReceipt>
- <dsig:Reference> whose URI attribute references a <dsig:Object> containing the <dsig:SignatureValue> of the signature being PostMarked (Sign) or Verified and PostMarked (Verify)

EPM-produced <PostMarkedReceipt>'s, always bind the receipt to the signature just created or verified.

Please refer to the EPM documentation for specific policy and usage guidelines.

```
<xs:element ref="epm:PostMarkedReceipt"

<!-- imported from the EPM schema -->
<xs:element name="PostMarkedReceipt" type="epm:PostMarkedReceiptType">

<xs:complexType name="PostMarkedReceiptType">
    <xs:sequence>
        <xs:element name="Receipt" type="epm:ReceiptType"/>
        <xs:element name="TimeStampToken" type="epm:QualifiedDataType" minOccurs="0"/>
        <xs:element name="ReceiptSignature" type="epm:QualifiedDataType" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>

```

```
<xs:complexType name="ReceiptType">
    <xs:sequence>
        <xs:element name="TransactionKey" type="epm:TransactionKeyType"/>
        <xs:element name="TSAX509SubjectName" type="xs:string"/>
        <xs:element name="TimeStampValue" type="xs:string"/>
        <xs:element name="RevocationStatusQualifier" type="xs:string"/>
        <xs:element name="ClientApplication" type="epm:ClientApplicationType"/>
        <xs:element name="ResourceID" type="xs:string"/>
    </xs:sequence>
</xs:complexType>
```

Note 1: The ReceiptSignature child element of the PostMarkedReceipt is only used when processing CMS/PKCS7 signatures where the receipt is standalone. It is simply used to protect the integrity of this standalone XML structure which contains an encapsulated CMS/PKCS7 <TimeStampToken>.

Note 2: The binary <TimeStampToken> element above can be omitted for XMLSig-based <SignatureType>'s since the PostMarkedReceipt is itself a signature which covers the <TstInfo> structure. EPM implementations using TimeStamp Authorities (TSAs), are however free to initialized this element with an RFC3161 timestamp token produced by a TSA if they wish. The example is section 6 does not initialize the <TimeStampToken> element.

2.7.2 Input / Output Element <TransactionKey>

This complexType is a compound key made up of 3 elements uniquely identifying each event in the an EPM Lifecycle. It is optionally specified on input when Lifecycle support is required, and always returned on output. The EPM generates and returns a new and unique <TransactionKey> with all response operations. This returned Key can be used to tie together events in a business workflow. When users or applications are adding another event to an existing lifecycle they need only supply that particular Key as part of their request. When they are referring to a particular event within the Lifecycle, then both the Key element and the Sequence element are required on input as part of the request. The <Locator> element is used to identify the particular EPM instance when multiple EPM instances are involved, as is the case with cross-border transactions. Please refer to EPM documentation for usage guidelines.

```
<xs:element ref="epm:TransactionKey"

<!-- imported from the EPM schema -->
<xs:element name=" TransactionKey" type="epm:TransactionKeyType">
<xs:complexType name="TransactionKeyType">
    <xs:sequence>
        <xs:element name="Locator" type="epm:LocatorType"/>
        <xs:element name="Key" type=" xs:string"/>
        <xs:element name="Sequence" type="xs:string"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="LocatorType">
    <xs:sequence>
        <xs:element name="CountryCode" type="xs:string"/>
            <xs:element name="Version" type="xs:string"/>
            <xs:element name="ServiceProvider" type="xs:string" nillable="true"/>
            <xs:element name="Environment" type="xs:string" nillable="true"/>
    </xs:sequence>
</xs:complexType>
```

2.7.3 Input Element <OrganizationID>

This element is used when the requester's organization name cannot be derived from the certificate. In those circumstances, this element should be initialized to the requester's organizational name as a xs:string. Please refer to EPM documentation for usage guidelines.

```
<xs:element name="OrganizationID" type="xs:string" nillable="true"/>
```

2.7.4 Input Element <ContentIdentifier>

This element is used to qualify the nature of the data being signed and is used by the EPM at retrieval time (e.g. RetrieveResults, RetrieveSummary, and Sign for Pickup) to both qualify searches for customer-specific content as well as providing access authorization categories. Please refer to EPM documentation for usage guidelines.

```
<xs:element name="ContentType" type="xs:string" nillable="true"/>
```

2.7.5 Input / Output Element <ClientApplicationID>

This element is an optional input and if initialized is echoed back as part of output responses. It is used to indicate to the EPM service the name of the client's application making the request. When results of previous operations are retrieved, this value is echoed back to the user. Please refer to EPM documentation for usage guidelines.

```
<xs:element name="ClientApplication" type="xs:string" nillable="true"/>
```

2.7.6 Input Element <ContentMetaData>

This element can be used by the user to provide any additional context surrounding the signing activity that they do not wish to have included in the signature itself. When results of previous operations are retrieved using the extended features of the EPM, this value is echoed back to the user. This element can also be used to refine subsequent customer pickup content searches in a free-form way. Please refer to EPM documentation for usage guidelines.

3 Profile of Signing Protocol

3.1 Element <SignRequest>

3.1.1 Constraints on Element <OptionalInputs>

Details on the constraints and semantics which exist with respect to the optional inputs as described in **[DSSCore]** follow in this section**.** All <OptionalInputs> not explicitly mentioned in this section are supported as defined in **[DSSCore]**.

EPM-specific <OptionalInputs> are described below in the section entitled EPM-specific <OptionalInputs>.

3.1.1.1 Element SignatureType

The <SignatureType> element MUST be included in the EPM profile's SignRequest.

The following <SignatureType> URNs are supported:

- Signature creation URNs:
  - urn:ietf:rfc:3275 (i.e. an XML Digital Signature)
  - urn:ietf:rfc:3369 (i.e. a CMS/PKCS7 binary Signature)
- Timestamp creation URNs:
  - oasis:names:tc:dss:1.0:core:schema:XMLTimeStampToken
  - urn:ietf:rfc:3161 (i.e. a CMS timestamp token)

The first 2 URNs instruct the EPM to create a signature as it's primary objectives. The last 2 URNs instruct the EPM to create a timestamp. The context and processing rules within which the EPM creates signatures is different than the context within which the EPM creates timestamps. These differences will be highlighted below as they apply to each optional input and output, as constrained by the <SignatureType> chosen above. If no restriction is mentioned below, one may assume that the optional input is valid for timestamp <SignatureType>'s as well. Specific processing for timestamp types is further described in section 3.2.4 entitled Timestamp Handling Profile of Sign Protocol.

3.1.1.2 Element <KeySelector>

The <KeySelector> optional input must be supported by the EPM, but is not required when calling the EPM service as a client user (i.e. is optional). If the EPM cannot derive the key to use for signing from the underlying authentication being used, or if the X509SubjectName is not readily available, the <KeySelector> can be used. When using EPM signing templates, users may initialize the <KeyInfo> element in the signing template with a valid X509SubjectName in the <KeyName> child element of <KeyInfo>. The EPM will utilize the specified certificate/key as defined. See Example 1 in section 5 for an example of signing templates.

Note: This optional input does not apply when users are requesting a timestamp <SignatureType>. EPM implementations are, by definition, TimeStamp Authorities and will use TSA-specific signing keys expressly for that purpose.

3.1.1.3 Element <SignedReferences> not Supported

The <SignedReferences> optional input element of the <SignRequest> as defined in **[DSSCore]** is not supported by EPM implementations. If a user requires greater control over signature references, they should use an EPM signing template included as part of the <DocumentWithTemplate> element.

EPM-supported signing templates contain not only the data to be signed, but also the format and directives on how the signature should be created, expressed as valid **[XMLSig]** elements. **[XMLSig]** elements such as <SignatureValue>, <DigestValue>, and <X509Certificate> are populated by the EPM service. The user simply leaves these elements empty, and the EPM service will automatically include the generated content and return the signed document as part of the <SignResponse>. See section 3.1.2.1 for details and section 5 for selected examples. More details are available in the EPM Systems Integrator's Guide and other EPM documentation available at the UPU's Postal Technology Centre http://www.ptc.upu.int/.

This signing template may contain as many valid <Reference> structures as are required. All <References> however must be resolved within the scope of the <DocumentWithTemplate> element. Users wishing to sign multiple XML node sets should include them all in the input <DocumentWithTemplate> element as children of the root. <SignedReferences> may be supported in a future version of the EPM profile. Please consult the OASIS DSS site for updates.

Note: This optional input is not supported and therefore does not apply when users are requesting a timestamp <SignatureType>.

3.1.1.4 Element <AddTimestamp>

The EPM supports this <OptionalInputs> element when used in the Sign protocol. Processing differs based on the <SignatureType> specified in the request. When specified, the EPM will create a timestamp and return this timestamp within the signature being created. In other words, this directive will result in the addition of a timestamp to the resultant signature. The Type attribute of the <AddTimestamp> element may be omitted since the EPM supports only "signature timestamps". That is to say the EPM will first generate the signature, and then embed the generated timestamp into that signature. How embedding takes place differs by <SignatureType>. The timestamps are added as follows for each <SignatureType> supported:

- For <SignatureType> values of urn:ietf:rfc:3369 (i.e. CMS/PKCS7), the timestamp will be produced as follows:
  - The following object identifier identifies the Signature Timestamp attribute:
  - id-aa-signatureTimeStampToken OBJECT IDENTIFIER ::=

{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 14 }

i.e. 1.2.840.113549.1.9.16.2.14

  - The value of messageImprint field within TimeStampToken shall be a hash of the value of signature field within SignerInfo for the signedData being timestamped.
  - The RFC 3161 timestamp token will be added as an unauthenticated attribute, as defined above, of the generated signature. The result will be returned in <SignatureObject>.
  - This approach is identical to that prescribed by ETSI TS 101 733 and is the convention utilized and supported by the EPM profile.
- For <SignatureType> values of urn:ietf:rfc:3275 (i.e. **[XMLSig]**), the timestamp will be produced as follows:
  - the timestamp will be produced as defined in section 5.1 of [DSSCore].
  - The URI attribute of the first <Reference> element of the timestamp's <SignedInfo> will point to an <Object> containing the <TstInfo> element.
  - The URI attribute of the second <Reference> element of the timestamp's <SignedInfo> will point to an <Object> containing the <SignatureValue> element of the signature just created. This is because the timestamp being added is a signature timestamp and not a content timestamp.
  - The detached timestamp signature will be included as the first child of the XML document root and will immediately precede the generated signature.
- The <AddTimestamp> optional input is not supported on timestamp-related <SignatureType>'s as used within the Signing protocol, since one cannot add a timestamp to a timestamp.

See also section 3.1.3.3 <IssuePostMarkedReceipt> which delivers different functionality than the <AddTimestamp> and results in the creation of an additional standalone <PostMarkedReceipt> structure which is a superset of a basic timestamp. Both optional inputs are supported on a Sign operation and serve different purposes.

On a Verify operation, the <ReturnUpdatedSignature> acts like the <AddTimestamp> described above, and will update the signature being verified. See also <IssuePostMarkedReceipt> which returns a timestamped receipt structure and has different semantic meaning. Please refer to section 4 covering the Verify.

Note: Content timestamps, created before signature generation, are currently not supported in the EPM profile (e.g. a pre and a post signature generation timestamp). They may however be added in a subsequent release of the EPM profile.

3.1.1.5 Element <InputDocuments>

The EPM profile constrains the <Document> element in that the dss:Base64Data choice is mandated when formatting the <Document> element. That is to say, that the dss:XMLData choice is not supported on input and never used on output. When manipulating XML content the EPM will employ a MimeType attribute value of "text/xml".

The EPM profile also constrains the <InputDocuments> element such that the EPM server presently accepts **only one** <Document> or <DocumentHash> element (i.e. equivalent of maxOccurs="1"). This may change in a subsequent version of the EPM profile. Users wishing to create signatures with multiple <Reference> elements should use EPM signing templates. See section 3.1.2.1 for details.

When the <Document> element is passed in by the user of this profile, it is assumed that it contains only the content to be signed. This is the standard convention used by **[DSSCore]** when passing in content to be signed and is supported by the EPM profile as the default. When users wish to use the EPM's "signing template" mechanism, they must pass the document template in on the <DocumentWithTemplate> element. Please refer to section 3.1.2.1 below.

The <Document> element is also used to pass in a signature to be timestamped when the <SignatureType> specifies a timestamp type. The MimeType should specify application/pkcs7-signature when passing in a signature to be timestamped.

Note: The <InputDocuments> element has special considerations when used in the context of timestamp-related <SignatureType>'s. Please refer to section 3.2.4 for more detailed explanation of the timestamping implications on use of the <InputDocuments> element.

3.1.1.6 Element <SignaturePlacement>

When creating RFC3275-compliant XML Digital Signatures in the Sign protocol as specified by the <SignatureType> optional input element, the <SignaturePlacement> optional input element is not required. Handling of signature placement is as follows.

- Enveloped Signatures (XMLSig)

  The EPM will assume an enveloped signature unless an <EnvelopingSignature> optional input is present. The resultant signature will be inserted after the last child of the root node of the <Document> element of <InputDocuments> and will be returned in the <DocumentWithSignature> element. This is default handling.

- Enveloping Signatures (XMLSig)

  For users requesting <EnvelopingSignature>'s the EPM will place the <Document> within a Reference'd <Object> element inside the signature and therefore requires the <Document>'s RefURI attribute in order to construct the <Reference>'s URI attribute as well as the corresponding <Object>'s Id attribute (without the # symbol). The result will be returned in the <SignatureObject> element as per **[DSSCore]**. If users wish to specify transforms and other signature features, they should use an EPM signing template. See section 3.1.2.1 for details.

- Enveloping Signatures (CMS/PKCS7)

  In order to obtain an enveloping CMS/PKCS7 signature, the caller must include the `<EnvelopingSignature>` optional input. When specified, the input document's content will be encapsulated in the CMS/PKCS7 SignedData signature structure and returned in the `<SignatureObject>` element. When the `<EnvelopingSignature>` optional input is specified, **only one** `<InputDocuments>` occurrence is supported (i.e. can only have a single child), and the `WhichDocument` attribute, if present, will be ignored.

- Detached Signatures (CMS/PKCS7)

  This is default processing for CMS/PKCS7 signatures and is identical to **[DSSCore]**. The input document's content will **NOT** be encapsulated in the CMS/PKCS7 SignedData signature structure as per detached signature conventions. This signature will be returned in the `<SignatureObject>` element.

- Same Document Detached Signatures (XMLSig)

  Callers can obtain detached XMLSig signatures by constructing a "signing template" reflecting a same document detached signature in the `<DocumentWithTemplate>` input element. A same document detached XMLSig signature will also be produced under an `<IssuePostMarkedReceipt>` scenario when the `Location` attribute is specified as `embedded`. See section 3.1.3.3. These signatures are returned in the `<DocumentWithSignature>` element.

This optional input does not apply when users are requesting a timestamp `<SignatureType>` in which case placement is determined by the server.

When greater and more direct control of signature placement is required, users must use an EPM signing template. EPM signing templates should be used if user wishes to have a same-document detached XMLSig-based signature created. See section 3.1.2.1 for details.

Note: As with **[DSSCore]**, **[XMLSig]** signatures created from `<DocumentHash>` will be returned in `<SignatureObject>`.

3.1.2 EPM-specific <OptionalInputs>

The following additional elements are specific to the EPM profile. There specific usage and constraints are documented below.

3.1.2.1 Element <DocumentWithTemplate>

The `<DocumentWithTemplate>` optional input element is used when users elect to utilize the EPM's "signing template" mechanism. EPM-supported signing templates contain not only the data to be signed, but also the format and directives of the signature to be created, expressed as valid **[XMLSig]** elements. In this fashion more elaborate signatures involving transforms, signed and unsigned properties, manifests, and multiple `<Reference>` elements can be supported. **[XMLSig]** elements such as `<SignatureValue>`, `<DigestValue>, and <X509Certificate>` are populated by the EPM service. The user simply leaves the element tags empty, and the EPM service will automatically include the generated content and return the signed document in the `<DocumentWithSignature>` element of the `<SignResponse>`. See Example 1 in section 5 for an example of signing templates. More details are available in the EPM Systems Integrator's Guide and other EPM documentation available at the UPU's Postal Technology Centre http://www.ptc.upu.int/.

```
<xs:element name="DocumentWithTemplate"> type="dss:Base64Data"/>
```

3.1.2.2 Element <TransactionKey>

Please refer to the description in section 2.7.2 entitled Element <TransactionKey>

3.1.2.3 Element <ClaimedIdentity>

This complexType is an extension of the standard OASIS DSS <ClaimedIdentity> element. This extension to ClaimedIdentity utilizes the OASIS <SupportingInfo> to define EPM-specific additions required to support the authentication and assertion of the requester's identity. The default authentication mechanism of an EPM implementation is external to the EPM profile and is supported by the conventions used in that underlying binding. In this fashion EPM implementations are free to authenticate users using standard approaches like HTTP Basic Authentication (i.e. `Authorization: Basic` in the HTTP header), or may decide to use stronger techniques involving Digest Authentication, encrypted cookies, one-time password schemes, two-factor tokens, or any of several other authentications schemes they chose. However there are situations where the underlying binding may not support the representation or the transport of the desired token type. For this reason, the EPM profile allows the chosen token type to be passed as "Authentication Information" as an attestation of, in support of, in addition to, or instead of the underlying authentication scheme and its assertion of identity. As such, it not used solely as additional authentication information, but rather could be used as an adjunct to the authentication mechanism itself. This scheme-specific authentication support is carried in the abstract <AlternateIdentity> type.

The <RequesterSignature> element is optional and is used in support of "Proof-of-Possession" or "Proof-of-Delivery" in the EPM's non-repudiation context. This element and its use-cases are further defined in the EPM Service Description documentation.

```
<xs:element name="ClaimedIdentity">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="Name" type="saml:NameIdentifierType"/>
            <xs:element ref="epm:SupportingInfo">
        </xs:sequence>
    </xs:complexType>
</xs:element>

<!-- imported from the EPM schema -->
<xs:element name="SupportingInfo" type="epm:SupportingInfoType"/>
<xs:complexType name="SupportingInfoType">
    <xs:element name="BasicAuth" type="epm:BasicAuthType" nillable="true"/>
    <xs:element name="RequesterSignature" type="epm:QualifiedDataType" nillable="true"/>
    <xs:element name="AlternateIdentity" type="epm:AlternateIdentityType" nillable="true"/>
</xs:complexType>

<xs:complexType name="epm:QualifiedDataType">
    <xs:simpleContent>
        <xs:extension base="xs:base64Binary">
            <xs:attribute name="MimeType" type="xs:string"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
```

3.1.2.4 Element <OrganizationID>

Please refer to the description in section 2.7.3 entitled Input Element <OrganizationID>

3.1.2.5 Element <ContentIdentifier>

Please refer to the description in section 2.7.4 entitled Input Element <ContentIdentifier>

3.1.2.6 Element <ClientApplicationID>

Please refer to the description in section 2.7.5 entitled Input / Output Element <ClientApplicationID>

3.1.2.7 Element <ContentMetaData>

Please refer to the description in section 2.7.6 entitled Input Element <ContentMetaData>

3.1.3 <OptionalInputs> Processing Flags

This section describes the <OptionalInputs> that are simple processing directives for the EPM. Each flag directs the EPM to perform specific functions and/or return specific response information. More detail on each processing option can be found in the EPM documentation.

3.1.3.1 Element <ExtendLifecycle>

Including this empty directive element instructs the EPM to extend the LifeCycle referred to in the <TransactionKey> element. Callers must initialize the <TransactionKey> sub-elements when using <ExtendLifecycle>.

```
<xs:element name="ExtendLifecycle"/>
```

3.1.3.2 Element <EndLifecycle>

Including this empty directive element instructs the EPM to close the LifeCycle and not allow any further events to be included in the LifeCycle identified by this <TransactionKey>.

```
<xs:element name="EndLifecycle"/>
```

3.1.3.3 Element <IssuePostMarkedReceipt>

Including this empty directive element instructs the EPM to return a signed receipt attesting to the origin of the signature as well as the validity of the certificate used in the signature process. Inclusion of this element results in the return of either a standalone <PostMarkedReceipt> signature containing its signed <PostMarkedReceipt> and <TimeStampToken> or one embedded in the signature being created. This element contains a Location attribute instructing the EPM how to return the <PostMarkedReceipt>. Processing differs based on the <SignatureType> and the value of the Location attribute.

- For a Location attribute value of standalone regardless of the <SignatureType>, processing is as follows:
  - The <PostMarkedReceipt> XML element will be returned as a standalone optional output structure as defined in section 2.7.1. Standalone <PostMarkedReceipt>'s are self-contained and contain a timestamp signature which binds the receipt to the signature value of the signature being created as part of this Sign operation.
- For a Location attribute value of embedded and a <SignatureType> value of urn:ietf:rfc:3275 (i.e. XMLSig), processing is as follows:
  - The EPM will first create an **[XMLSig]** based "detached" signature covering the input document. The input document's contents will be outside the produced signature and referenced by it. The EPM will then add a <PostMarkedReceipt> detached signature structure covering the <SignatureValue> of the first signature just created. The resulting signed and PostMarked document will be returned in the <DocumentWithSignature> element.
- A Location attribute value of embedded with a <SignatureType> value of urn:ietf:rfc:3369 (i.e. CMS/PKCS7) is not supported.
  - A signature timestamp (i.e. an RFC 3161 timestamptoken) however can be embedded in a CMS/PKCS7 signature by using the <AddTimestamp> optional input described in section 3.1.1.4. This timestamp bears the Issuer name of the Post's TimeStamp Authority.

Please refer to section 6 for a detailed example of a <PostMarkedReceipt> signature.

```
<xs:element name="IssuePostMarkedReceipt">
    <xs:complexType>
        <xs:attribute name="Location" type="xs:string"/>
    </xs:complexType>
```

```
</xs:element>
```

3.1.3.4 Element <StoreNonRepudiationEvidence>

Including this empty directive element instructs the EPM to store evidence of the operation being performed. This evidentiary information can be subsequently retrieved and used to support challenges as to its authenticity.

```
<xs:element name="StoreNonRepudiationEvidence"/>
```

3.1.3.5 Element <ReturnSignatureInfo>

Including this empty directive element instructs the EPM to return additional response information relating to the signing operation. This directive element results in the return of a <SignatureInfo> structure in the <OptionalOutputs>.

```
<xs:element name="ReturnSignatureInfo"/>
```

3.1.3.6 Element <ReturnX509Info>

Including this empty directive element instructs the EPM to return additional response information relating to the certificate used for the signing. This directive element results in the return of an <X509Info> structure in the <OptionalOutputs>.

```
<xs:element name="ReturnX509Info"/>
```

3.2 Element <SignResponse>

3.2.1 Element <Result>

This profile defines an additional <ResultMajor> code as follows:

urn:oasis:names:tc:dss:1.0:resultmajor:Warning

All EPM result codes are always accompanied by a <ResultMessage> element.

3.2.2 Element <SignatureObject>

If successful, the server will return a <dsig:Signature> with the signature properties as defined in **[DSSCore].** Location of the generated signature will be determined based on signature type and envelope type. All CMS/PKCS7 signatures will be returned in the <SignatureObject> element. XMLSig based enveloping signatures will also be returned in the <SignatureObject> element. See also section 3.1.1.6 entitled Element <SignaturePlacement>.

Note: Special cases of the <SignatureObject> as an output element exist when specifying timestamp <SignatureType>'s and is described further in section 3.2.4.3.

3.2.3 EPM-specific <OptionalOutputs>

The following additional elements are specific to the EPM profile. There specific usage and constraints are documented below.

3.2.3.1 Element <TransactionKey>

Please refer to section 3.1.2.2 for a description of how the <TransactionKey> element is used on both input and on output as both an identification mechanism and to support the concept of a multi-event LifeCycle.

3.2.3.2 Element <PostMarkedReceipt>

If the <IssuePostMarkedReceipt> optional input is included, then this optional output will be returned. It is essentially a standalone receipt represented as an enveloping signature. See section 2.7.1 for details.

3.2.3.3 Element <DocumentWithSignature>

This element will initialized and returned for XMLSig based signatures. Additionally If the caller is using EPM signing templates and has passed in a signing template (See section 3.1.2.1) in the <DocumentWithTemplate> element, then this output element will contain the signed document. See also section 3.1.1.6 entitled Element <SignaturePlacement>.

3.2.3.4 Element <SignatureInfo>

This structure can be returned on both the Sign and Verify operations and is returned whenever the <ReturnSignatureInfo> element is included in the request. Together with the <X509Info> element these elements provide more detail on the signature just created or being verified. The element <SignedContent> is used in conjunction with the Verify operation and allows users to extract the signed content from a CMS signature thus alleviating the need to parse the ASN.1 structure. See <VerifyResponse> below. The <SignedContent> element on a CMS Sign response is empty as the user has just passed this content in to be signed, however the <SignedContent> element on an XMLDSig Sign request will contain the transformed content as it existed prior to digest calculation for the user's reference.

Detailed explanation of the other <SignatureInfo> elements can be found in the EPM System Integrator's Guide.

```
<xs:element ref="epm:SignatureInfo"

<!-- imported from the EPM schema -->
<xs:element name="SignatureInfo" type="epm:SignatureInfoType">
<xs:complexType name="SignatureInfoType">
    <xs:sequence>
        <xs:element name="SignedContent" type="base64Binary" nillable="true"/>
        <xs:element name="ContentHash" type="xs:string"/>
        <xs:element name="ContentHashAlgo" type="xs:string"/>
        <xs:element name="ContentEncryptAlgo" type="xs:string"/>
        <xs:element name="SigningTime" type="xs:string"/>
        <xs:element name="PKCS1" type="epm:QualifiedDataType" nillable="true"/>
    </xs:sequence>
</xs:complexType>
```

Note: This optional output does not apply when users have requested a timestamp <SignatureType>.

3.2.3.5 Element <X509Info>

This structure is returned whenever the <ReturnX509Info> element is included in the request. The <X509ValidationData> element contains the signed OCSP Validation Data (or equivalent) returned by the EPM attesting to the validity of the certificate used in the signing operation. It will contain signed content as per RFC 2560 and also described in RFC 3126 as returned by a standard OCSP Responder. If an EPM DSS implementation is not using an OCSP responder, then sufficient certificate chain and revocation references must be included here. Additionally, many jurisdictions (e.g. the EU) require that this validation info be signed by the Certified Service Provider (CSP). This is not an issue when using an RFC 2560-compliant OCSP Responder. Definitions of the other elements can be found in the EPM Systems Integrator's Guide.

```
<xs:element ref="epm:X509Info"

<!-- imported from the EPM schema -->
<xs:element name="X509Info" type="epm:X509InfoType">
<xs:complexType name="X509InfoType">
    <xs:sequence>
        <xs:element name="X509Subject" type="xs:string"/>
        <xs:element name="X509Issuer" type="xs:string"/>
        <xs:element name="X509Serial" type="xs:string"/>
        <xs:element name="X509StatusSource" type="xs:string"/>
        <xs:element name="X509ValidFrom" type="xs:string"/>
        <xs:element name="X509ValidTo" type="xs:string"/>
        <xs:element name="X509Certificate" type="xs:string"/>
        <xs:element name="X509RevocationReason" type="xs:string" nillable="true"/>
        <xs:element name="X509RevocationReason" type="xs:string" nillable="true"/>
        <xs:element name="X509RevocationTime" type="xs:string" nillable="true"/>
        <xs:element name="X509ValidationData" type="xs:base64Binary" nillable="true"/>
    </xs:sequence>
</xs:complexType>

```

Note: This optional output does not apply when users have requested a timestamp <SignatureType>.

3.2.4 Timestamp Handling Profile of Sign Protocol

When users are requesting a timestamp <SignatureType> using the EPM Sign profile, input to the timestamp operation can be either of several formats depending on the user's requirements. Currently the following 3 options for creating a timestamp token using the EPM's Sign profile are supported. Please note that this explicit request for a timestamp token is different than the EPM's PostMark and receipting mechanism which can be used on both Sign and Verify operations, and carries specific semantics and functional support. It is also different from the <IssuePostMarkedReceipt> option on the Verify which allows for timestamp and receipt embedding.

3.2.4.1 Standalone Timestamp from <DocumentHash>

This timestamp creation scenario is supported by the EPM exactly as described in the Timestamping Profile of the Sign Protocol. The <SignatureType> will govern the type of timestamp to be created. Supported values are:

- oasis:names:tc:dss:1.0:core:schema:XMLTimeStampToken
- urn:ietf:rfc:3161

The timestamp will be returned in a DSS <SignatureObject> of either of the following types:

- <dss:Timestamp> of type RFC3161TimeStampToken
- <dss:Timestamp> of type <dsig:Signature> formatted as an XML Timestamp Token as per section 5.1.1 of **[DSSCore]**

3.2.4.2 Standalone Timestamp from <Document>

This timestamp creation scenario is similar to 3.2.4.1 except that the EPM will calculate the hash for the user as a courtesy function, prior to creating and returning the timestamp. As such it's input and output is identical to 3.2.4.1 above, with the one exception that the data to be used as input to the hash calculation is provided on an incoming <Document> element instead of a <DocumentHash> element. The EPM implementation should decide on the hash algorithm to be employed based on its own TSA Policy.

3.2.4.3 Embedding a Signature Timestamp into a user-provided Signature

This timestamp creation scenario is used under the Sign protocol when the user has an "existing" signature object in their possession and wishes to have it timestamped as specified in the incoming <SignatureType>. The timestamp will be embedded in the user's signature object and is handled as follows:

- For <SignatureType> values of urn:ietf:rfc:3161
  - The EPM detects from the MimeType attribute (of the <Base64Data> element of the <Document> element) that the input is a CMS/PKCS7 signature. The EPM will embed an RFC 3161 timestamp token into the incoming signature and return the resultant CMS/PKCS7 in the <SignatureObject> element. The timestamp will be embedded as an unsigned attribute as specified in section 3.1.1.4
- For <SignatureType> values of oasis:names:tc:dss:1.0:core:schema:XMLTimeStampToken
  - Users must pass in an XMLSig compliant signature in the <Document> input element. There must be at most a single signature in the <Document> element and the signature value within that signature must be specified as <dsig:SignatureValue>. The EPM will return a <dss:timestamp> as a standalone <dsig:Signature> in the <SignatureObject> element. The <dss:timestamp> will be very similar to Example 1 in section 6 except that the 2nd <Reference> will not be present. The user must splice the returned <dss:timestamp> into the signed document as desired. It should be noted that timestamping in this manner works best when timestamping detached signatures so as not to invalidate the original signature after splicing the timestamp signature back into the original document.

It should be noted that no verification is performed on the incoming signature prior to timestamping. A signature timestamp calculated over the signature value of the incoming signature and the subsequent embedding of that timestamp into the incoming signature is all that is performed. For timestamping of verified signatures, please refer to the Verify Protocol below.

4 Profile of Verifying Protocol

4.1 Element <VerifyRequest>

4.1.1 Constraints on Element <OptionalInputs>

4.1.1.1 Element <SignatureObject>

Must be initialized when users wish to verify CMS/PKCS7 based signatures, or XMLSig based enveloping signatures which contain the signed content.

4.1.1.2 Element <InputDocuments>

Must be initialized when users wish to verify XMLSig based enveloped signatures which contain the signed content. Presently constrained to one `<Document>` occurrence. This single `<Document>` must contain signature(s) to be verified as well as all signed content referenced by the signature(s).

4.1.1.3 Element <VerifyManifests>

This optional input element is not supported by the EPM.

4.1.1.4 Element <VerificationTime>

This optional input element is not supported by the EPM. Users should utilize any of the supported timestamp or PostMark facilities when official time is required.

4.1.1.5 Element <AdditionalKeyInfo>

This optional input element is not required by the EPM.

4.1.1.6 Element <ReturnProcessingDetails>

This optional input element is not supported by the EPM.

4.1.1.7 Element <ReturnSigningTime>

This optional input element is not required by EPM implementations as this information is returned to the caller in the `<SignatureInfo>` element which can be optionally requested by including the `<ReturnSignatureInfo>` optional input.

4.1.1.8 Element <ReturnSignerIdentity>

This optional input element is not required by the EPM as this information is returned in the `<X509Info>` element which can be optionally requested by including the `<ReturnX509Info>` optional input.

4.1.1.9 Element <ReturnUpdatedSignature>

This optional input is only valid when verifying CMS/PKCS7 signatures and allows for the embedding of conventional binary RFC 3161 timestamp tokens into the incoming signature after successful verification. It produces an embedded timestamp similar to the one produced as part of the Sign protocol and described in section 3.1.1.4 entitled Element <AddTimestamp>. The RFC3161 complaint timestamp token is included in the signature as an unauthenticated attribute of the verified signature. This conventionally timestamped CMS/PKCS7 signature, now updated, will be returned in the `<SignatureObject>`.

4.1.1.10 Element <ReturnTransformedDocument>

This optional input element is not supported by the EPM.

4.1.2 EPM-specific <OptionalInputs>

4.1.2.1 Element <TransactionKey>

See section 3.1.2.2 for a detailed explanation of this elements usage..

4.1.2.2 Element <OrganizationID>

See section 3.1.2.4 for a detailed explanation of this elements usage.

4.1.2.3 Element <ContentIdentifier>

See section 3.1.2.5 for a detailed explanation of this elements usage.

4.1.2.4 Element <ClientApplicationID>

See section 3.1.2.6 for a detailed explanation of this elements usage.

4.1.2.5 Element <ContentMetaData>

See section 3.1.2.7 for a detailed explanation of this elements usage.

4.1.3 <OptionalInputs> Processing Flags

This section describes the `<OptionalInputs>` that are simple processing directives for the EPM. Each flag directs the EPM to perform specific functions and/or return specific response information. More detail on each processing option can be found in the EPM documentation.

4.1.3.1 Element <ExtendLifecycle>

See section 3.1.3.1 for a detailed explanation of this elements usage.

4.1.3.2 Element <EndLifecycle>

See section 3.1.3.2 for a detailed explanation of this elements usage.

4.1.3.3 Element NodeName

The optional `<NodeName>` element qualifies the signature(s) to be verified by the EPM. If the user wishes to Verify a particular signature or signatures, they can be included in `<NodeName>` element(s). This element may also serve useful if the user in unsure of exactly what has been verified, and wishes to control the verification process more explicitly.

```
<xs:element name="NodeName" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
```

In the example below, the user would specify string values of `lgl:Party1` and/or `lgl:Party2` to explicitly instruct the EPM what to Verify. By default the EPM will search for signature nodes specified as `<dsig:Signature>`, which appear as descendants of the document root.

```
<lgl:Party1>
   <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
   ...
   </dsig:Signature>
</lgl:Party1>
<lgl:Party2>
   <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
   ...
   </dsig:Signature>
</lgl:Party2>
```

4.1.3.4 Element <IssuePostMarkedReceipt>

This optional input instructs the EPM to issue a PostMarkedReceipt signature as attestation of successful verification of the incoming signature(s). A <PostMarkedReceipt> signature will not be returned if the incoming signature(s) do not verify successfully or the revocation status of the public verification certificate is not zero.

When specifying this element on a Verify operation, the EPM will use a <NodeName> element if it is present. The <PostMarkedReceipt> will cover the signature(s) that have been verified.

Processing differs based on the <SignatureType> and the value of the Location attribute.

- For a Location attribute value of standalone regardless of the <SignatureType>, processing is as follows:
  - The <PostMarkedReceipt> XML element will be returned as a standalone optional output structure as defined in section 2.7.1. Standalone <PostMarkedReceipt>'s are self-contained and contain a timestamp signature which binds the receipt to the signature value of the signature being verified as part of this Verify operation.
- For a Location attribute value of embedded and a <SignatureType> value of urn:ietf:rfc:3275 (i.e. XMLSig), the incoming <Document> containing the signature(s) **must** be a **detached** XMLSig based signature. Processing is as follows:

  The incoming signed document will contain an **[XMLSig]** based "detached" signature covering the required content within the input document. The input document's signed content will be outside the signature and referenced by it. The EPM will verify this signature. If the signature(s) verify successfully, the EPM will then add a <PostMarkedReceipt> detached signature structure covering the <SignatureValue>'s of the signature(s) just verified.
  - The resulting PostMarked document will be returned in the <DocumentWithSignature> element and will include the <PostMarkedReceipt> attesting to its validity.
- A Location attribute value of embedded with a <SignatureType> value of urn:ietf:rfc:3369 (i.e. CMS/PKCS7) is not supported.
  - A signature timestamp (i.e. an RFC 3161 timestamptoken) however can be embedded in a CMS/PKCS7 signature by using the <AddTimestamp> optional input described in section 3.1.1.4. This timestamp bears the Issuer name of the Post's TimeStamp Authority.

Please refer to section 6 for a detailed example of a <PostMarkedReceipt> signature.

```
<xs:element name="IssuePostMarkedReceipt">
    <xs:complexType>
        <xs:attribute name="Location" type="xs:string"/>
    </xs:complexType>
</xs:element>
```

4.1.3.5 Element <StoreNonRepudiationEvidence>

See section 3.1.3.4 for a detailed explanation of this elements usage.

4.1.3.6 Element <ReturnSignatureInfo>

See section 0 for a detailed explanation of this elements usage.

4.1.3.7 Element <ReturnX509Info>

See section 3.1.3.6 for a detailed explanation of this elements usage.

4.2 Element <VerifyResponse>

4.2.1 Element <Result>

This profile defines an additional <ResultMajor> code as follows:

urn:oasis:names:tc:dss:1.0:resultmajor:Warning

All EPM result codes are always accompanied by a <ResultMessage> element.

4.2.2 Element <SignatureObject>

This element is only returned when the <ReturnUpdatedSignature> optional input is included. Please refer to section 4.1.1.9 for details.

4.2.3 Element <OptionalOutputs>

4.2.3.1 Element <DocumentWithSignature>

If the <IssuePostMarkedReceipt> optional input is included and its Location attribute specifies embedded, then this optional output will be returned. See the scenario described in the 2nd bullet within section 4.1.3.3 above for more details.

4.2.4 Element <EPM-specific OptionalOutputs>

The following additional elements are specific to the EPM profile. There specific usage and constraints are documented below.

4.2.4.1 Element <TransactionKey>

Please refer to section 3.1.2.2 for a description of how the <TransactionKey> element is used on both input and on output as both an identification mechanism and to support the concept of a multi-event LifeCycle.

4.2.4.2 Element <PostMarkedReceipt>

If the <IssuePostMarkedReceipt> optional input is included in the Verify request and its Location attribute specifies standalone, then this optional output will be returned. It is essentially a standalone receipt signature. See also section 4.1.3.3 above.

4.2.4.3 Element <SignatureInfo>

See section 0 for a detailed explanation of this elements usage.

4.2.4.4 Element <X509Info>

See section 3.2.3.5 for a detailed explanation of this elements usage.

5 Signing Template Examples

This section reproduces a few illustrative Sign template examples from the EPM Signature generation service. For full details on features and options of the EPM XML Digital Signature signing templates, please consult the UPU EPM System Integrator's Guide.

Example 1:

This first example is a simple enveloped signature template which uses the standard enveloped-signature transform and the illustrated digest method. Note how the `<SignatureValue>` element is simply left empty. The EPM Service will expand all valid empty element tags with appropriate content. This particular example also requests that selected `<X509Data>` elements be completed. This is accomplished by including empty `<X509Certificate>`, `<X509SubjectName>`, and `<X509IssuerSerial>` elements.

```
<?xml version="1.0" encoding="UTF-8"?>
<DocumentWithTemplate>
   <Data>
      <SubData1>
         <SubSubData1 MimeType="text/plain">This is the data to be signed.</SubSubData1>
         <SubSubData2 MimeType="text/plain">This is the data to be signed.</SubSubData2>
         <SubSubData3 MimeType="text/plain">This is the data to be signed.</SubSubData3>
      </SubData1>
      <SubData2>This is the data to be signed.</SubData2>
      <SubData3>This is the data to be signed.</SubData3>
   </Data>
   <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
      <SignedInfo>
         <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
         <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
            <Reference URI="">
               <Transforms>
                  <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
               </Transforms>
               <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
               <DigestValue></DigestValue>
            </Reference>
      </SignedInfo>
      <SignatureValue>
      </SignatureValue>
      <KeyInfo>
         <KeyName>C=CA, S=Ontario, L=Ottawa, O=CPC, OU=eServices, CN=Ed Test, E=ed.shallow@rogers.com</KeyName>
         <X509Data>
            <X509Certificate></X509Certificate>
            <X509SubjectName></X509SubjectName>
            <X509IssuerSerial>
            </X509IssuerSerial>
```

```
            </X509Data>
        </KeyInfo>
    </Signature>
</DocumentWithTemplate>
```

Example 2:

This example is similar to the first however an Xpointer is used within the <Reference> element's URI attribute. This approach is useful when specific subsets of the document require signing. Again certificate information is added to the produced signature.

```
<?xml version="1.0" encoding="UTF-8"?>
<DocumentWithTemplate>
    <Data>
        <SubData1>
            <SubSubData1 MimeType="text/plain">This is the data to be signed.</SubSubData1>
            <SubSubData2 MimeType="text/plain">This is the data to be signed.</SubSubData2>
            <SubSubData3 MimeType="text/plain">This is the data to be signed.</SubSubData3>
        </SubData1>
        <SubData2>This is data.</SubData2>
        <SubData3>This is data.</SubData3>
    </Data>
    <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
        <SignedInfo>
            <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
            <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
                <Reference URI="#xpointer(/DocumentWithTemplate/Data/SubData1)">
                    <Transforms>
                        <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                    </Transforms>
                    <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                    <DigestValue></DigestValue>
                </Reference>
        </SignedInfo>
        <SignatureValue>
        </SignatureValue>
        <KeyInfo>
            <X509Data>
                <X509Certificate></X509Certificate>
                <X509SubjectName></X509SubjectName>
                <X509IssuerSerial>
                </X509IssuerSerial>
            </X509Data>
        </KeyInfo>
    </Signature>
</DocumentWithTemplate>
```

Example 3:

This is a more complicated example using intersect and subtract XPath Filters. This 3rd example illustrates step 1 in a multi-party contract signing workflow. This template controls the scope of data to be signed by the first party. A similar template would be used by the second party after the first party has signed the document. This second template would simply change the “subtract” value in the transform filter. Again certificate information is added to the produced signature.

```
<?xml version="1.0"?>
<DocumentWithTemplate>
   <Contract>
      <Terms MimeType="text/plain">This is the data to be signed by both parties</Terms>
      <Conditions MimeType="text/plain">This is the data to be signed by both parties</Conditions>
      <Obligations MimeType="text/plain">This is the data to be signed by both parties</Obligations>
      <Party1>
         <Terms MimeType="text/plain">This is the data to be signed by party 1</Terms>
         <Conditions MimeType="text/plain">This is the data to be signed by party 1</Conditions>
         <Obligations MimeType="text/plain">This is the data to be signed by party 1</Obligations>
      </Party1>
      <Party2>
         <Terms MimeType="text/plain">This is the data to be signed by party 2</Terms>
         <Conditions MimeType="text/plain">This is the data to be signed by party 2</Conditions>
         <Obligations MimeType="text/plain">This is the data to be signed by party 2</Obligations>
      </Party2>
   </Contract>
   <dsig:Signature xmlns:dsig-xpath="http://www.w3.org/2002/06/xmldsig-filter2">
      <dsig:SignedInfo>
         <dsig:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
         <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
         <dsig:Reference URI="">
            <dsig:Transforms>
               <dsig:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
               <dsig:Transform Algorithm="http://www.w3.org/2002/06/xmldsig-filter2">
                  <dsig-xpath:XPath Filter="intersect">//Contract</dsig-xpath:XPath>
                  <dsig-xpath:XPath Filter="subtract">//Party2</dsig-xpath:XPath>
               </dsig:Transform>
            </dsig:Transforms>
            <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <dsig:DigestValue></dsig:DigestValue>
         </dsig:Reference>
      </dsig:SignedInfo>
      <dsig:SignatureValue>
      </dsig:SignatureValue>
      <dsig:KeyInfo>
         <dsig:X509Data>
            <dsig:X509Certificate></dsig:X509Certificate>
            <dsig:X509SubjectName></dsig:X509SubjectName>
            <dsig:X509IssuerSerial></dsig:X509IssuerSerial>
         </dsig:X509Data>
```

```
        </dsig:KeyInfo>
    </dsig:Signature>
</DocumentWithTemplate>
```

6 PostMarkedReceipt Examples

PostMarked receipts are normally returned to the application as standalone XML structures, whether they are of type CMS/PKCS7 or XMLSig. Upon request however <PostMarkedReceipt>'s can be embedded in the incoming signed document. This is true for both the Sign protocol as well as the Verify protocol. The first example below is a standalone <PostMarkedReceipt>, and the second example is one that is embedded into the signed document.

Example 1:

This is an example of a PostMarkedReceipt. It is essentially a conventional XMLSig enveloping signature over the <SignatureValue> of the target signature being PostMarked. It contains three (3) <Reference> elements pointing to each of the following:

- a standard <dss:TstInfo> as per **[DSSCore]**
- an <epm:PostMarkedReceipt> element from the **[EPM]** schema
- the <SignatureValue> element of the target signature being PostMarked

Selected element contents have been deliberately truncated for brevity and clarity.

```
<?xml version="1.0" encoding="UTF-8"?>
<dsig:Signature Id="PostMarkedReceipt_001" xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
   <dsig:SignedInfo>
      <dsig:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
      <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
      <dsig:Reference URI="#TstInfo040327174718Z">
         <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
         <dsig:DigestValue>jWkUFR6epvkrtaxTiQ33DiWy+l8=</dsig:DigestValue>
      </dsig:Reference>
      <dsig:Reference URI="#Receipt040327174718Z">
         <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
         <dsig:DigestValue>9JWKdLh/8Cs9Slu2QmZixOJl+x0=</dsig:DigestValue>
      </dsig:Reference>
      <dsig:Reference URI="#PostMarkedSignature">
         <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
         <dsig:DigestValue>MOBYPfrllMBcJz6yojbhrwH9KP4=</dsig:DigestValue>
      </dsig:Reference>
   </dsig:SignedInfo>
   <dsig:SignatureValue>qnBvJoSgo4OoiYYaE3AwbL5/EDq7BhTT6 ... Qw11HK+zxy66I=</dsig:SignatureValue>
   <dsig:KeyInfo>
      <dsig:KeyName>C=CA, O=CPC, OU=EPM Service, CN=EPM Signature, E= … </dsig:KeyName>
      <dsig:X509Data>
      <X509Certificate xmlns="http://www.w3.org/2000/09/xmldsig#">MIIEUDC … EwZOBg==</X509Certificate>
<X509SubjectName xmlns="http:// … xmldsig#"> C=CA, O=CPC, OU=EPM Service, CN=EPM Signature, E=… </X509SubjectName>
      <X509IssuerSerial xmlns="http://www.w3.org/2000/09/xmldsig#">
         <X509IssuerName>C=CA, O=CPC, OU=EPM Service, CN=Electronic PostMark CA, E=… </X509IssuerName>
```

```
            <X509SerialNumber>25</X509SerialNumber>
         </X509IssuerSerial>
         </dsig:X509Data>
      </dsig:KeyInfo>
      <dsig:Object xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
         <dss:TstInfo xmlns:dss="http://www.docs.oasis-open.org/dss/2004/06/oasis-dss-1.0-core-schema-wd-30.xsd"
                      Id="TstInfo040327174718Z">
            <SerialNumber>1847365279</SerialNumber>
            <CreationTime>2004-03-27T17:47:18.750</CreationTime>
            <Policy/>
            <ErrorBound/>
            <Ordered/>
            <TSA>C=CA, O=CPC, OU=EPM Service, CN=EPM Signature, E= … </TSA>
         </dss:TstInfo>
      </dsig:Object>
      <dsig:Object xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
         <epm:PostMarkedReceipt xmlns:epm="http://www.upu.int/EPMService" Id="Receipt040327174718Z">
            <Receipt>
               <TransactionKey>
                  <Locator>CA</Locator>
                  <Key>1234567890</Key>
                  <Sequence>1</Sequence>
               </TransactionKey>
               <ServiceProviderURL>http://epmproxy.avalonworks.com/upubeta/</ServiceProviderURL>
               <TSAX509SubjectName>C=CA, O=CPC, OU=EPM Service, CN=EPM Signature, … </TSAX509SubjectName>
               <TimeStampValue>040327174718Z</TimeStampValue>
               <RevocationStatusQualifier>CRL Checked</RevocationStatusQualifier>
               <ResourceID>12345678</ResourceID>
            </Receipt>
            <TimeStampToken MimeType="application/pkcs7-signature"/>
            </epm:PostMarkedReceipt>
      </dsig:Object>
      <dsig:Object xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
         <epm:PostMarkedSignatureValue xmlns:epm="http://www.upu.int/EPMService"
               Id="PostMarkedSignature">Vif2Y7aShziFOCy0sDUOR9XVnVCy8LW9hY ... 2RsmQETPmsM=</epm:PostMarkedSignatureValue>
      </dsig:Object>
   </dsig:Signature>
```

Note: Similarly, when the <PostMarkedReceipt>'s signature scope simply covers data, as when used as described in section 3.2.4.2, then the 3rd <Reference> will be to an <Object> containing the hash of the data to be PostMarked with base64 encoding specified.

```
<dsig:Object xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
```

```
   <PostMarkedData Id="PostMarked"
Encoding="http://www.w3.org/2000/09/xmldsig#base64">RGF0YSBJIHdhbnQgdG8gcQ...gVGVzdCBNYXkgMTUgMTI6MTA=</PostMarkedData>
</dsig:Object>
```

Example 2:

This is an example of an embedded <PostMarkedReceipt> returned after a successful Verify operation. It is a conventional XMLSig detached signature over the <SignatureValue> of the target signature(s) being PostMarked. It contains three (3) <Reference> elements pointing to each of the following:

- a standard <dss:TstInfo> as per **[DSSCore]**
- an <epm:PostMarkedReceipt> element from the **[EPM]** schema
- the <SignatureValue> element of the target signature(s) being PostMarked

Note that depending on the value of the optional NodeName element specified on the <IssuePostMarkedReceipt> within the request, the <PostMarkedReceipt> can potentially cover all <SignatureValue>'s in the signed document when the document contains multiple signatures.

Selected element contents have been deliberately truncated for brevity and clarity.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Document [
<!ATTLIST Object Id ID #IMPLIED>
]>
<Document>
<!-- Beginning of PostMarkedReceipt signature -->
   <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
      <dsig:SignedInfo>
         <dsig:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
         <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
         <dsig:Reference URI="#TstInfo040327174718Z">
            <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <dsig:DigestValue>3Lk/6TE71dqeXZFUJ9qqaPInm24=</dsig:DigestValue>
         </dsig:Reference>
         <dsig:Reference URI="#Receipt040327174718Z">
            <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <dsig:DigestValue>430zTvcoa9r8Rpr5DiVZf7IPvl8=</dsig:DigestValue>
         </dsig:Reference>
         <dsig:Reference URI="">
            <dsig:Transforms>
               <dsig:Transform Algorithm="http://www.w3.org/TR/1999/REC-xslt-19991116">
                  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                                  xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" version="1.0">
                     <xsl:template match="/">
                        <xsl:copy>
```

```
                              <xsl:copy-of select="//dsig:Signature[position()!=1]/dsig:SignatureValue"/>
                           </xsl:copy>
                        </xsl:template>
                     </xsl:stylesheet>
                  </dsig:Transform>
               </dsig:Transforms>
               <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
               <dsig:DigestValue>LRAX6mCfAq8hprb8UMU1H35PTYw=</dsig:DigestValue>
            </dsig:Reference>
         </dsig:SignedInfo>
         <dsig:SignatureValue>qnBvJoSgo4OoiYYaE3AwbL5/EDq7BhTT6 ... Qw11HK+zxy66I=</dsig:SignatureValue>
         <dsig:KeyInfo>
            <dsig:KeyName>C=CA, O=CPC, OU=EPM Service, CN=EPM Signature, E= … </dsig:KeyName>
            <dsig:X509Data>
            <X509Certificate xmlns="http://www.w3.org/2000/09/xmldsig#">MIIEUDC … EwZOBg==</X509Certificate>
      <X509SubjectName xmlns="http:// … xmldsig#"> C=CA, O=CPC, OU=EPM Service, CN=EPM Signature, E=… </X509SubjectName>
            <X509IssuerSerial xmlns="http://www.w3.org/2000/09/xmldsig#">
               <X509IssuerName>C=CA, O=CPC, OU=EPM Service, CN=Electronic PostMark CA, E=… </X509IssuerName>
               <X509SerialNumber>25</X509SerialNumber>
            </X509IssuerSerial>
            </dsig:X509Data>
         </dsig:KeyInfo>
         <dsig:Object xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
            <dss:TstInfo xmlns:dss="http://www.docs.oasis-open.org/dss/2004/06/oasis-dss-1.0-core-schema-wd-25.xsd"
                         Id="TstInfo040327174718Z">
               <SerialNumber>1847365279</SerialNumber>
               <CreationTime>2004-03-27T17:47:18.750</CreationTime>
               <Policy/>
               <ErrorBound/>
               <Ordered/>
               <TSAX509SubjectName>C=CA, O=CPC, OU=EPM Service, CN=EPM Signature, … </TSAX509SubjectName>
            </dss:TstInfo>
         </dsig:Object>
         <dsig:Object xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
            <epm:PostMarkedReceipt xmlns:epm="http://www.upu.int/EPMService" Id="Receipt040327174718Z">
               <Receipt>
                  <TransactionKey>
                     <Locator>CA</Locator>
                     <Key>1234567890</Key>
                     <Sequence>1</Sequence>
                  </TransactionKey>
                  <ServiceProviderURL>http://epmproxy.avalonworks.com/upubeta/</ServiceProviderURL>
                  <TSAX509SubjectName>C=CA, O=CPC, OU=EPM Service, CN=EPM Signature, … </TSAX509SubjectName>
                  <TimeStampValue>040327174718Z</TimeStampValue>
                  <RevocationStatusQualifier>CRL Checked</RevocationStatusQualifier>
                  <ResourceID>12345678</ResourceID>
               </Receipt>
               <TimeStampToken MimeType="application/pkcs7-signature"/>
               </epm:PostMarkedReceipt>
```

```
        </dsig:Object>
    </dsig:Signature>
<!-- End of PostMarkedReceipt signature -->
<!-- Beginning of signed document being PostMarked -->
    <Object Id="DetachedDataBeingSigned">
        <PersonalData>
            <Name>Ed Shallow</Name>
            <StreetAddress>1234 Mockingbird Lane</StreetAddress>
            <City>Yellowknife</City>
            <PostalCode>W1C6J3</PostalCode>
            <SocialInsuranceNumber>123456789</SIN>
        </PersonalData>
    </Object>
    <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" Id="TargetSignature">
        <dsig:SignedInfo>
            <dsig:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
            <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
            <dsig:Reference URI="#DetachedDataBeingSigned">
                <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                <dsig:DigestValue>Po3vwPXh8kdpRUAzMGjzluao65I=</dsig:DigestValue>
            </dsig:Reference>
        </dsig:SignedInfo>
        <dsig:SignatureValue>KyKUMJKW ... Yi7swX0FjLkDDZNs=</dsig:SignatureValue>
        <dsig:KeyInfo>
            <dsig:KeyName>C=CA, O=Acme Corp, CN=Joe Public, E= … </dsig:KeyName>
            <dsig:X509Data>
            <X509Certificate xmlns="http://www.w3.org/2000/09/xmldsig#">MIIE … EwZOBg==</X509Certificate>
            <X509SubjectName> C=CA, O=Acme Corp, CN=Joe Public, E= … </X509SubjectName>
            <X509IssuerSerial>
                <X509IssuerName>C=CA, O=Partner CA, O=For Test Use Only, CN=Partner CA, E= … </X509IssuerName>
                <X509SerialNumber>25</X509SerialNumber>
            </X509IssuerSerial>
            </dsig:X509Data>
        </dsig:KeyInfo>
    </dsig:Signature>
<!-- End of signed document being PostMarked -->
</Document>
```

7 Element cross-reference Table

The following tables provide a summary of the Input elements, options, and corresponding Output elements for each of the usage scenarios. Comments are also provided.

Sign Protocol

	Optionality	As Used in Sig Type		Affected Elements	Comments
		CMS/PKCS7	XMLSIG		
Input / Request Elements					
TransactionKey	O	✓	✓	The same element is returned as part of the SignResponse and contains the unique identifier. This unique key may be passed in on subsequent calls to tie events together, when doing so, users should initialize the `<ExtendLifecycle>` optional input element.	
OrganizationID	M	✓	✓		Must match the string specified at registration time.
ContentIdentifier	O	✓	✓		Optionally specified, and is used at retrieval time to filter access to signed and verified content of a particular type.
ClientApplicationID	O	✓	✓		Optionally specified by the client and used to identify operations which originated from a particular application or desktop software product.
ContentMetaData	O	✓	✓		Additional context data related to the Sign or Verify operation.
SignatureType	M	✓	✓		Tells the EPM whether this is a sign request, or a timestamp request, and also specifies CMS/PKCS7 or

						XMLSig.
	KeySelector	O	✓	✓		Optional since the key can usually can be derived from the underlying authentication mechanism. Also not required when using signing templates, in which case the key may be specified in `<KeyInfo>`. Can be used when non-default handling is required.
	SignedReferences	n/a				Not required by the EPM. This functionality is covered by signing templates in the EPM Profile.
	InputDocuments	M	✓	✓		Presently constrained to one `<Document>` occurrence.
	SignaturePlacement	n/a				Not required. Default handling of placement is supported by the EPM. Signature placement can be controlled as required by using signing templates. Placement of `<PostMarkedReceipt>`'s can be controlled via the `Locator` attribute.
	DocumentWithTemplate	O		✓	Signatures produced will be returned in `<DocumentWithSignature>`	When users are passing in XMLSig signing templates, they must be placed here.
	ClaimedIdentity	O	✓	✓		Optionally used for alternate authentication schemes or when "Proof of Delivery" is required.
Processing Option Flags						
	AddTimestamp	O	✓			Attribute not req'd. Produces a conventional timestamp as opposed to a `<PostMarkedReceipt>`.
	ExtendLifecycle	O			Users must initialize `<TransactionKey>` when using this optional flag.	
	EndLifecycle	O			Closes off a multiple event	

					Lifecycle.	
	IssuePostMarkedReceipt	O	✓		Returns a `standalone` `<PostMarkedReceipt>` element.	
	IssuePostMarkedReceipt	O		✓	Returns a `standalone` `<PostMarkedReceipt>` element in the response if the `Location` attribute is specified as `standalone`. If `Location` specifies `embedded`, the receipt will be embedded and returned in `<DocumentWithSignature>`.	
	StoreNonRepudiationEvidence	O				The EPM will log the original request as well as the response and all result structures as evidence in the event of a dispute.
	ReturnSignatureInfo	O	✓	✓	Returns a `<SignatureInfo>` structure.	
	ReturnX509Info	O	✓	✓	Returns a `<X509Info>` structure.	
Output / Response Elements						
	Result	M	✓	✓		As per **[DSSCore]**. `<ResultMajor>`, `<ResultMinor>`, and `<ResultMessage>` will all be initialized and returned
	TransactionKey	M	✓	✓		Always initialized and returned.
	SignatureObject	M	✓	✓	Initialized for CMS/PKCS7 signatures and for XMLSig enveloping signatures. See also `<DocumentWithSignature>`	
	DocumentWithSignature	O		✓	Only initialized for XMLSig based enveloped and detached signatures produced with PostMarkedReceipts. See also	

					<SignatureObject>.	
	PostMarkedReceipt	O	✓	✓	See <IssuePostMarkedReceipt> above.	
	SignatureInfo	O	✓	✓	Returned when <ReturnSignatureInfo> has been specified.	
	X509Info	O	✓	✓	Returned when <ReturnX509Info> has been specified.	

1243

1244 **Verify Protocol**

		Optionality	As Used in Sig Type		Elements Affected	Comments
			CMS/PKCS7	XMLSIG		
Input / Request Elements						
	TransactionKey	O	✓	✓	The same element is returned as part of the VerifyResponse and contains the unique identifier. This unique key may be passed in on subsequent calls to tie events together, when doing so, users should initialize the <ExtendLifecycle> optional input element.	
	OrganizationID	M	✓	✓		Must match the string specified at registration time.
	ContentIdentifier	O	✓	✓		Optionally specified, and is used at retrieval time to filter access to signed and verified content of a particular type.
	ClientApplicationID	O	✓	✓		Optionally specified by the client and used to identify operations which originated from a particular application or desktop software product.

	ContentMetaData	O	✓	✓		Additional context data related to the Sign or Verify operation.
	SignatureObject	M	✓			Required when verifying CMS/PKCS7 signatures.
	InputDocuments	O	✓			Required when verifying CMS/PKCS7 detached signatures, in which case both this element and the `SignatureObject` above must be initialized.
	SignatureObject	M		✓		Required when verifying XMLSig enveloping signatures which contain the signed content.
	InputDocuments	M		✓		Presently constrained to one `<Document>` occurrence. Must contain signature(s) to be verified along with any referenced signed content.
	ClaimedIdentity	O	✓	✓		Optionally used for alternate authentication schemes or when “Proof of Delivery” is required.
Processing Option Flags						
	ReturnUpdatedSignature	O	✓		Updated CMS/PKCS7 signature, now containing an embedded RFC 3161 timestamp token, is returned in `<SignatureObject>`.	Allows for the inclusion of an RFC3161 embedded timestamp into the verified CMS/PKCS7 signature.
	ExtendLifecycle	O			Users must initialize `<TransactionKey>` when using this optional flag.	
	EndLifecycle	O			Closes off a multiple event Lifecycle.	
	IssuePostMarkedReceipt	O	✓		Returns a `standalone <PostMarkedReceipt>` element.	
	IssuePostMarkedReceipt	O		✓	Returns a `standalone <PostMarkedReceipt>`	

					element in the response if the `Location` attribute is specified as `standalone`. If `Location` specifies `embedded`, the receipt will be embedded and returned in `<DocumentWithSignature>`. The `NodeName` element optionally controls the scope of the `PostMarkedReceipt` signature.	
	StoreNonRepudiationEvidence	O				The EPM will log the original request as well as the response and all result structures as evidence in the event of a dispute.
	ReturnSignatureInfo	O	✓	✓	Returns a `<SignatureInfo>` structure.	
	ReturnX509Info	O	✓	✓	Returns a `<X509Info>` structure.	
Output / Response Elements						
	Result	M	✓	✓		As per **[DSSCore]**. `<ResultMajor>`, `<ResultMinor>`, and `<ResultMessage>` will all be initialized and returned
	TransactionKey	M	✓	✓		Always initialized and returned.
	PostMarkedReceipt	O	✓	✓	See `<IssuePostMarkedReceipt>` above.	
	SignatureObject	O	✓		Initialized for CMS/PKCS7 signatures when `<ReturnUpdatedSignature>` has been specified. See also `<DocumentWithSignature>`	
	DocumentWithSignature	O		✓	Only initialized for XMLSig based signatures when `<IssuePostMarkedReceipt>` with a `Location` attribute specified as `embedded`. See also	

					<IssuePostMarkedReceipt>.	
	SignatureInfo	O	✓	✓	Returned when <ReturnSignatureInfo> has been specified.	
	X509Info	O	✓	✓	Returned when <ReturnX509Info> has been specified.	

1245

1246

8 References

8.1 Normative

[Core-XSD] T. Perrin et al. *DSS Schema*. OASIS, (MONTH/YEAR TBD)

[DSSCore] T. Perrin et al. *Digital Signature Service Core Protocols and Elements*. OASIS, **(MONTH/YEAR TBD)**

[RFC 2119] S. Bradner. Key words for use in RFCs to Indicate Requirement Levels. IETF RFC 2396, August 1998.

http://www.ietf.org/rfc/rfc2396.txt.

[TS 101733] Advanced Electronic Signatures. ETSI TS 101 733.

[XAdES] XML Advanced Electronic Signatures. ETSI TS 101 903, February 2002 (shortly to be reissued).

[XML-ns] T. Bray, D. Hollander, A. Layman. *Namespaces in XML.* W3C Recommendation, January 1999.

http://www.w3.org/TR/1999/REC-xml-names-19990114

[XMLSig] D. Eastlake et al. *XML-Signature Syntax and Processing.* W3C Recommendation, February 2002.

http://www.w3.org/TR/1999/REC-xml-names-19990114

[RFC 2634] P. Hoffman (ed.). Enhanced Security Services for S/MIME, June 1999.

[RFC 3369] Message Syntax (CMS). R. Housley. August 2002.

[EPM] Universal Postal Union, Electronic PostMark Web Service Description Language (WSDL)

the UPU's Postal Technology Centre http://www.ptc.upu.int/.

1265 # Appendix A. Revision History

Rev	Date	By Whom	What
wd-01	2004-07-27	Ed Shallow	Initial version
wd-02	2004-08-18	Ed Shallow	Update
wd-03	2004-09-06	Ed Shallow	Update
wd-04	2004-10-14	Ed Shallow	Juan-Carlos and Trevor's changes
wd-05	2004-11-21	Ed Shallow	Changes for Public Draft
wd-06	2004-11-30	Ed Shallow	Changes for Public Draft

Appendix B. Notices

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification, can be obtained from the OASIS Executive Director.

OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director.