Electronic PostMark (EPM) Profile of the OASIS Digital Signature Service Version 1.0

OASIS Standard

11 April 2007

Specification URIs:

This Version:

http://docs.oasis-open.org/dss/v1.0/oasis-dss-profiles-epm-spec-v1.0-os.html

http://docs.oasis-open.org/dss/v1.0/oasis-dss-profiles-epm-spec-v1.0-os.pdf

http://docs.oasis-open.org/dss/v1.0/oasis-dss-profiles-epm-spec-v1.0-os.doc

Latest Version:

http://docs.oasis-open.org/dss/v1.0/oasis-dss-profiles-epm-spec-v1.0-os.html

http://docs.oasis-open.org/dss/v1.0/oasis-dss-profiles-epm-spec-v1.0-os.pdf

http://docs.oasis-open.org/dss/v1.0/oasis-dss-profiles-epm-spec-v1.0-os.doc

Technical Committee:

OASIS Digital Signature Services TC

Chair(s):

Nick Pope, Thales eSecurity
Juan Carlos Cruellas, Centre d'aplicacions avançades d'Internet (UPC)

Editor(s):

Ed Shallow, Universal Post Union

Related work:

This specification is related to:

- oasis-dss-core-spec-v1.0-os

Abstract:

This document 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 document was last revised or approved by the membership of OASIS Digital Signature Services TC on the above date. The level of approval is also listed above. Check the current location noted above for possible later revisions of this document. This document is updated periodically on no particular schedule.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at http://www.oasis-open.org/committees/dss/.

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 Technical Committee web page (http://www.oasis-open.org/committees/dss/ipr.php.

The non-normative errata page for this specification is located at http://www.oasis-open.org/committees/dss/.

Notices

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, receipting, and evidence logging 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.

Implementers wishing to take their implementations of this profile to market are asked to do so through any of the Postal Administrations participating or wishing to participate in the global EPM initiative. Any client is free to develop service request calls which adhere to this interface and receive their corresponding service responses.

1.1 Terminology

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 Normative References

[Core-XSD] S. Drees et al. *DSS Schema*. OASIS, February 2007

[DSSCore] S. Drees et al. *Digital Signature Service Core Protocols and Elements*. OASIS, February 2007

[RFC 2396] S. Bradner. Key words for use in RFCs to Indicate Requirement Levels. http://www.ietf.org/rfc/rfc2396.txt, IETF RFC 2396, August 1998. .

[TS 101733] CMS Advanced Electronic Signatures. ETSI TS 101 733, January 2007.

[XAdES] XML Advanced Electronic Signatures. ETSI TS 101 903, March 2006

[XML-ns] T. Bray, D. Hollander, A. Layman. *Namespaces in XML.* http://www.w3.org/TR/1999/REC-xml-names-19990114, W3C Recommendation, January 1999.

[XMLSig] D. Eastlake et al. *XML-Signature Syntax and Processing.* http://www.w3.org/TR/1999/REC-xml-names-19990114, W3C Recommendation, February 2002.

[RFC 2634] P. Hoffman (ed.). Enhanced Security Services for S/MIME, http://www.ietf.org/rfc/rfc2634.txt, IETF RFC 2634 June 1999.

[RFC 3369] R. Housley, Message Syntax (CMS).. http://www.ietf.org/rfc/rfc3369.txt , IETF RFC 3369 August 2002.

[EPM] Universal Postal Union, Electronic PostMark Web Service Description Language (WSDL) the UPU's Postal Technology Centre http://www.ptc.upu.int/

1.3 Non-Normative References

1.4 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/2004/04/oasis-dss-1.0-profiles-EPM-wd-09#
```

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. One of the primary intents of the EPM Profile is to simplify request and response processing for client callers by constraining **[DSSCore]** in several ways.

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
- Same document signatures is the default and preferred mechanism for handling signature creation and verification
- Certificate validation data
    - Revocation references
    - Certificate references
    - Online Certificate Status Protocol (OCSP) responses
- Timestamping from a CA-independent TimeStamp Authority

This profile constrains the `<InputDocuments>` element in that it may contain only one `<Document>` element. Additionally, this profile assumes that documents and their signatures are to be contained in the same XML document wherever possible. This considerably simplifies the amount of splicing that a client must perform when dealing with the protocol. This will be evident in the Input and Output constraints explained herein.

2.3 Relationship To Other Profiles

The profile in this document is based directly on the **[DSSCore]**.

2.4 Signature Objects

This profile supports the creation and verification of XMLSig and CMS/PKCS7 signatures and timestamps as defined in **[DSSCore]**.

2.5 Transport Binding

This profile is transported using either an XML-based HTTP payload POSTed to an implementation of this profile, 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**.** Although outside the scope of this protocol, clients are expected to authenticate to an implementation of this specification. At a minimum HTTP Basic Authorization should be used to authenticate. Implementations are expected to validate the user and password contained in the HTTP header.

2.7 Common Elements

This section describes elements used and referenced within both the Sign and Verify protocols as either Input or Output elements.

2.7.1.1 Element <InputDocuments>

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. Multiple `<Reference>` elements are supported. 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 or the signed document to be verified. When users wish to use the EPM's "signing template" mechanism, they must also pass in a `<DocumentContainsTemplate>` element directive. Please also refer to section 3.1.2.1 below.

On the Verify protocol the processing differs slightly from the core in that input documents containing "same document" signatures can be passed in on a Verify request via the `Document` element. This avoids having to use the `SignatureObject` in conjunction with the `SignaturePtr` choice of that element. Since only one occurrence of the `Document` element is allowed, `SignaturePtr` is not required.

The `dss:TransformedData` choice is not supported by this profile.

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 RFC 3161 timestamped.

2.7.1.2 Element <DocumentWithSignature>

This element is used in conjunction with the `SignatureObject` element for returning signed and verified documents. For Sign operations, this element will be initialized and returned for **[XMLSig]** based signatures when the signature is an enveloped or detached one. Additionally if the caller is using EPM signing templates and has passed in a signing template (See section 3.1.2.1) by specifying the `DocumentContainsTemplate` element, then this output element will contain the signed document.

For verify operations this output element is only initialized for **[XMLSig]** based signatures when the `<IssuePostMarkedReceipt>` option is specified with a `Location` attribute specified as `embedded`. In this case the signed and verified document is returned along with the embedded `PostMarkedReceipt` in this output element. See also `<IssuePostMarkedReceipt>`.

```
<xs:element name="DocumentWithSignature">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="dss:Document"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

2.7.2 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 when the StoreNonRepudiationInfo Optional Input is specified.

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 additional 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:choice>
			<xs:element name="PKCS7SignedReceipt" type="epm:PKCS7SignedReceiptType"/>
			<xs:element name="XMLSignedReceipt" type="epm:QualifiedDataType"/>
		</xs:choice>
	</xs:sequence>
</xs:complexType>

<xs:complexType name="PKCS7SignedReceiptType">
	<xs:sequence>
		<xs:element name="Receipt" type="epm:ReceiptType"/>
		<xs:element name="ReceiptSignature" type="epm:QualifiedDataType"
nillable="true"/>
	</xs:sequence>
```

```
</xs:complexType>
<xs:complexType name="ReceiptType">
        <xs:sequence>
                <xs:element name="TransactionKey" type="epm:TransactionKeyType"/>
                <xs:element name="Requester" type="xs:string"/>
                <xs:element name="Operation" type="xs:string"/>
                <xs:element name="TSAX509SubjectName" type="xs:string"/>
                <xs:element name="TimeStampValue" type="xs:string"/>
                <xs:element name="RevocationStatusQualifier" type="xs:string"/>
                <xs:element name="TimeStampToken" type="epm:QualifiedDataType" nillable="true"
minOccurs="0" maxOccurs="1"/>
                <xs:element name="MessageImprint" type="xs:base64Binary" nillable="true"/>
                <xs:element name="PostMarkImage" type="epm:QualifiedDataType" nillable="true"/>
                <xs:element name="ReceiptMetadata" type="epm:ReceiptMetadataType"
nillable="true" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
</xs:complexType>

<xs:complexType name="ReceiptMetadataType">
        <xs:sequence>
                <xs:element name="Name" type="xs:string"/>
                <xs:choice>
                        <xs:element name="Value" type="xs:string"/>
                        <xs:element name="EncodedValue" type="epm:QualifiedDataType"/>
                <xs:choice>
        </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 initialize this element with an RFC3161 Timestamp Token if they wish. The example is section 6 does not initialize the `<TimeStampToken>` element.

2.7.3 Output Element <TransactionKey>

This complexType is a compound key made up of 3 elements uniquely identifying each event in the an EPM Lifecycle. The EPM generates and returns a new and unique `<TransactionKey>` with all response operations. 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:positiveInteger"/>
    </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>
```

2.7.4 Input Element <OrganizationID>

This element is used when the requester’s organization name cannot or should not be derived from a public certificate (as would be the case with X509 Mutual Authentication). In those circumstances, this element should be initialized to the requester’s organizational name as an `xs:string`. This value will be validated at authentication time by the EPM service against registration-time information.

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

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/PKCS7 timestamp token)

The first 2 URNs instruct the EPM to create a signature. 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.

3.1.1.2 Element <KeySelector>

The `<KeySelector>` optional input must be supported by EPM implementations of this profile, 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 <AddTimestamp>

The EPM supports this `<OptionalInputs>` element when used in the Sign protocol. Processing is the same as that described in **[DSSCore]**.

See also section 3.1.3.1 `<IssuePostMarkedReceipt>` which delivers similar but different functionality than the `<AddTimestamp>` and results in the creation of an additional standalone `<PostMarkedReceipt>` structure which is a superset of a basic dss:XMLTimestamp. Both optional inputs are supported on a Sign operation and serve different purposes.

The `<AddTimestamp>` is also supported on the Verify operation, 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 timestamp created before signature generation and referenced as a signed property or attribute). They may however be added in a subsequent release of the EPM profile.

3.1.1.4 Optional Input <Properties>

This optional input element is not supported by this profile. If specialized signed or unsigned properties are required users are encouraged to use the EPM's "signing templates" facility.

3.1.1.5 Optional Input <SignedReferences>

This optional input element is not supported by this profile. If greater control over `Reference` element creation is required, users are encouraged to use the EPM's "signing templates" facility.

3.1.1.6 Optional Input <IncludeObject>

This optional input is supported by the EPM profile to produce enveloping signatures with the following constraints.The `WhichDocument` attribute is not required and hence not supported. The `hasObjectTagsAndAttributesSet` is also not supported. This profile supports only one `<IncludeObject>` in the request. The default and only behavior supported in this profile for this optional input is exactly as is described in the `createReference` attribute as specified in **[DSSCore]**.

3.1.1.7 Optional Input <SignaturePlacement>

This element is supported with the following constraints. Only the last attribute of this element from **[DSSCore]** is supported. That is, the `CreateEnvelopedSignature` attribute is the only attribute supported. Default signature placement in this profile for enveloped signatures is to place the `ds:Signature` as the last child of the input Document's root.

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 <DocumentContainsTemplate>

The `<DocumentContainsTemplate>` optional input element is a directive which tells the implementation that the `<Document>` element passed in on the request is a "signing template". It 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 without complex XML request constructs. **[XMLSig]** elements such as `<SignatureValue>`, `<DigestValue>, and <X509Certificate>` are populated by the EPM service based on the template provided. The user leaves these crypto-specific 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 though the UPU.

Note: When using templates, all **[XMLSig]** References must resolve within the single `<Document>` element passed in on the request. This compromise was chosen to provide maximum flexibility and ease-of-use.

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

3.1.2.2 Element <TransactionKey>

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

3.1.2.3 Element <ClaimedIdentity>

This optional 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 is 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 available through the Universal Postal Union.

```
<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>

<xs:complexType name="BasicAuthType">
      <xs:sequence>
            <xs:element name="UserID" type="xs:string"/>
            <xs:element name="Password" type="xs:string" nillable="true"/>
      </xs:sequence>
</xs:complexType>

<xs:complexType name="AlternateIdentityType" abstract="true">
      <xs:sequence>
            <xs:element name="IdentityToken" type="xs:anyType"/>
      </xs:sequence>
</xs:complexType>
```

3.1.2.4 Element <OrganizationID>

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

3.1.3 <OptionalInputs> Processing Directives

This section describes the `<OptionalInputs>` that are simple processing directives for the EPM. Each directive or “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 <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 2.7.2. 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 `3.1.1.3.` 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 ref="epm:IssuePostMarkedReceipt">

<!-- imported from the EPM schema -->
<xs:element name="IssuePostMarkedReceipt" type="epm:IssuePostMarkedReceiptType">
<xs:complexType> name="IssuePostMarkedReceiptType">
    <xs:sequence>
        <xs:element name="Location" type="epm:ValidLocation" minOccurs="0"/>
        <xs:element name="PostMarkImage" type="epm:PostMarkImageType" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="PostMarkImageType">
    <xs:simpleContent>
        <xs:extension base="xs:boolean">
            <xs:attribute name="Format" type="xs:string" default="JPG"/>
            <xs:attribute name="Size" type="epm:ValidImageSize" default="Small"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
```

3.1.3.2 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"/>
```

3.1.3.3 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"/>
```

3.1.3.4 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 `<ds: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. Enveloped and detached signatures are returned in the `<DocumentWithSignature>` element described below.

3.2.3 EPM-specific <OptionalOutputs>

The following additional elements are specific to the EPM profile. Their 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 which is used on both input and on output as an identification and retrieval mechanism and to support subsequent reference to specific service calls.

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 2.7.2 for details.

3.2.3.3 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/PKCS7 signature thus alleviating the need to parse the ASN.1 structure. See `<VerifyResponse>` below. The `<SignedContent>` element on a CMS/PKCS7 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="epm:QualifiedDataType" nillable="true"/>
              <xs:element name="ContentHash" type="xs:string" nillable="true"/>
              <xs:element name="ContentHashAlgo" type="xs:string" nillable="true"/>
              <xs:element name="ContentEncryptAlgo" type="xs:string" nillable="true"/>
              <xs:element name="SigningTime" type="xs:string" nillable="true"/>
              <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>`.

3.2.3.4 Element <X509Info>

This structure is returned whenever the `<ReturnX509Info>` element is included in the request. The `<X509ValidationData>` element is the default implementation of the abstract base type called `GenericValidationDataType` and contains a signed OCSP Validation Data element returned by the EPM. This element attests 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 possibly as an alternate implementation of the abstarct base type. 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 are standard certificate fields and descriptions are also available 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" nillable="true"/>
       <xs:element name="X509Serial" type="xs:string" nillable="true"/>
       <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" nillable="true"/>
        <xs:element name="X509RevocationReason" type="xs:string" nillable="true"/>
        <xs:element name="X509RevocationReasonString" type="xs:string" nillable="true"/>
        <xs:element name="X509RevocationTime" type="xs:string" nillable="true"/>
        <xs:element name="X509ValidationData" type="epm:X509ValidationDataType"
nillable="true"/>
        </xs:sequence>
</xs:complexType>

<xs:complexType name="GenericValidationDataType" abstract="true">
        <xs:sequence>
                <xs:element name="GenericValidationData" type="xs:anyType"/>
        </xs:sequence>
</xs:complexType>

<xs:complexType name="X509ValidationDataType">
        <xs:complexContent>
                <xs:extension base="epm:GenericValidationDataType">
                        <xs:sequence>
                                <xs:element name="X509ValidationData"
type="epm:QualifiedDataType"/>
                        </xs:sequence>
                </xs:extension>
        </xs:complexContent>
</xs:complexType>
```

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

4 Profile of Verifying Protocol

4.1 Element <VerifyRequest>

4.1.1 Constraints on Element <OptionalInputs>

4.1.1.1 Element <InputDocuments>

Must be initialized when users wish to verify **[XMLSig]** based enveloped or detached 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) as part of a “same document” signature.

4.1.1.2 SignatureObject

Since “same document” signatures are common place and `InputDocuments` can be used as an input element on a Verify, the `SignatureObject` input element is not required on the Verify operation by this profile.

4.1.1.3 Element <AdditionalKeyInfo>

This optional input element is not required by the EPM.

4.1.1.4 Element <ReturnProcessingDetails>

This optional input element is not supported by the EPM.

4.1.1.5 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.

4.1.1.6 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.

4.1.1.7 Element <VerifyManifests>

This optional input element is not supported by the EPM. By default, the EPM verifies Manifest references and returns results in the same fashion as normal References. The EPM has a separate and related optional input which allows callers to suppress Manifest verification. See below.

4.1.1.8 Element <ReturnUpdatedSignature>

This optional input is not supported by the EPM. The only signature update supported is when the caller specifies `<AddTimestamp>`. This produces an embedded timestamp similar to the one produced as part of the Sign protocol and described in section 3.1.1.3 entitled Element <AddTimestamp>. The RFC3161 compliant 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>`.

4.1.1.9 Element <ReturnTransformedDocument>

This optional input element is not supported by the EPM.

4.1.2 EPM-specific <OptionalInputs>

4.1.2.1 Element <OrganizationID>

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

4.1.2.2 Element <IgnoreManifests>

This EPM-specific optional input allows callers to specify that Manifests be ignored during verification processing.

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

4.1.2.3 Element <SignatureSelector>

This optional `SignatureSelector` element qualifies the XMLDSIG signature(s) to be verified by the EPM. 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.

If the user wishes to Verify a particular signature or signatures, they have two choices has to how they may specify the `dsig:Signature` nodes to be verified. Each choice is a sub-element of the `SignatureSelectorType` below.

The **First** method allows users to specify any ancestor (parent) node of the signature(s) to be verified and are specified by including these names as `NodeName` element(s). The value is expressed as a string. A namespace URI qualifier may precede the actual signature `NodeName value`.

.

```
<xs:element name="SignatureSelector" type="epm:SignatureSelectorType"/>
    <xs:complexType name="SignatureSelectorType">
        <xs:sequence>
            <xs:choice>
                <xs:element name="NodeName" type="xs:string" minOccurs="1"
maxOccurs="unbounded"/>
                <xs:element name="XPathSelector" type="epm:XPathSelectorType"/>
            </xs:choice>
             </xs:sequence>
    </xs:complexType>

    <xs:complexType name="XPathSelectorType">
        <xs:sequence>
            <xs:element name="XPath" type="xs:string"/>
            <xs:element name="NameSpace" type="xs:string" nillable="true" />
            <xs:element name="Qualifer" type="xs:string" nillable="true"/>
        </xs:sequence>
    </xs:complexType>
```

EXAMPLE 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>
```

The **Second** method involves specifying an XPath expression which when evaluated will return the target `<dsig:Signature> nodes to be verified. The actual Xpath expression is included in the XPath element and any required namespace and qualifier can be specified in the NameSpace and Qualifier elements.`

EXAMPLE Using an XPath expression to select the target `<dsig:Signature>` *nodes*

```
<lgl:Document xmlns:lgl="http://www.lgl.org/SomeService"
             xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
      <lgl:Signatures>
             <dsig:Signature>1st</dsig:Signature>
             ...
             <dsig:Signature>2nd</dsig:Signature>
             ...
             <dsig:Signature>3rd</dsig:Signature>
             ...
      </lgl:Signatures>
</lgl:Document>
```

In the example above a value of `//lgl:Signatures//dsig:Signature[position=2]` would select only the second signature to be verified.

A value of `//lgl:Signatures//dsig:Signature` in the XPath element would cause all signatures to be verified.

In both examples a value of `http://www.lgl.org/SomeService and lgl` should be specified for the NameSpace and Qualifier elements respectively in order to allow the XPath string expression to evaluate.

4.1.2.4 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 `<SignatureSelector>` 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 2.7.2. 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 `3.1.1.3.` 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 ref="epm:IssuePostMarkedReceipt">

<!-- imported from the EPM schema -->
<xs:complexType> name="IssuePostMarkedReceiptType">
        <xs:sequence>
                <xs:element name="Location" type="epm:ValidLocation" minOccurs="0"/>
                <xs:element name="PostMarkImage" type="epm:PostMarkImageType" minOccurs="0"/>
        </xs:sequence>
</xs:complexType>

<xs:complexType name="PostMarkImageType">
        <xs:simpleContent>
                <xs:extension base="xs:boolean">
                        <xs:attribute name="Format" type="xs:string" default="JPG"/>
                        <xs:attribute name="Size" type="epm:ValidImageSize" default="Small"/>
                </xs:extension>
        </xs:simpleContent>
</xs:complexType>
```

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 <StoreNonRepudiationEvidence>

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

4.1.3.2 Element <ReturnSignatureInfo>

See section 3.1.3.3 or a detailed explanation of this elements usage.

4.1.3.3 Element <ReturnX509Info>

See section 3.1.3.4 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 `<AddTimestamp>` optional input is included. Please refer to section 4.1.1.8 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 4.1.2.4 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 0 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 3.2.3.4 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/>
<Document>
   <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/2001/10/xml-exc-c14n#"/>
         <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>
</Document>
```

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/>
<Document>
   <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/2001/10/xml-exc-c14n#"/>
         <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
            <Reference URI="#xpointer(/Document/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>
</Document>
```

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/>
<Document>
   <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/2001/10/xml-exc-c14n#"/>
         <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>
</Document>
```

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 Standalone PostMarkedReceipt:

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="PostMarkedReceiptSignature" xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
   <dsig:SignedInfo>
      <dsig:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
      <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
      <dsig:Reference URI="#TstInfo">
         <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
         <dsig:DigestValue>jWkUFR6epvkrtaxTiQ33DiWy+l8=</dsig:DigestValue>
      </dsig:Reference>
      <dsig:Reference URI="#Receipt">
         <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
         <dsig:DigestValue>9JWKdLh/8Cs9Slu2QmZixOJl+x0=</dsig:DigestValue>
      </dsig:Reference>
      <dsig:Reference URI="#PostMarkedSignatures">
         <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="urn:oasis:names:tc:dss:1.0:core:schema" Id="TstInfo">
            <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#" Id="Receipt">
         <epm:PostMarkedReceipt xmlns:epm="http://www.upu.int/EPMService/schemas">
            <Receipt>
               <TransactionKey>
                  <Locator>
                     <CountryCode>CA</CountryCode>
                     <Version>114</Version>
                     <ServiceProvider>ePost Corporation</ServiceProvider>
                     <Environment xsi:nil="true"/>
                  </Locator>
                  <Key>1234567890</Key>
                  <Sequence>1</Sequence>
               </TransactionKey>
               <Requester>CN=Joe Public, O=VeriSign Class 1 Certificate, C=CA, E=joe.public@rogers.com</Requester>
               <Operation>Verify</Operation>
               <TSAX509SubjectName> … </TSAX509SubjectName>
               <MessageImprint> … </MessageImprint>
               <PostMarkImage> … </PostMarkImage>
               <RevocationStatusQualifier>CRL Checked</RevocationStatusQualifier>
               <TimeStampToken MimeType="application/pkcs7-signature"></TimeStampToken>
               <ReceiptMetadata>
                  <Name> … </Name>
                  <Value>… </Value>
               </ReceiptMetadata>
            </Receipt>
         </epm:PostMarkedReceipt>
      </dsig:Object>
      <dsig:Object xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
         <epm:PostMarkedContent xmlns:epm="http://www.upu.int/EPMService/schemas" Id="PostMarkedSignatures">
             <epm:PostMarkedSignatureValue>1NiHC2bBKfT ... AlfhecQo=</epm:PostMarkedSignatureValue>
         </epm:PostMarkedContent>
      </dsig:Object>
   </dsig:Signature>
```

If the standalone PostMarkedReceipt covers more than one signature, the 3rd Referenced Object would look like this:

```
<dsig:Object xmlns:dsig=http://www.w3.org/2000/09/xmldsig# Id="PostMarkedSignatures">
    <epm:PostMarkedContent xmlns:epm="http://www.upu.int/EPMService/schemas">
        <PostMarkedSignatureValue>1NiHC2bBKfT ... AlfcGhecQo=</PostMarkedSignatureValue>
        <PostMarkedSignatureValue>aqw95gB/Tz5 ... n0qRqMHJ5c=</PostMarkedSignatureValue>        ...
        ... would include as many other PostMarkedSignatureValue elements as may be present in the PostMarked document
        ...
    </epm:PostMarkedContent>
</dsig:Object>
```

Note: Similarly, when the `<PostMarkedReceipt>`'s signature scope simply covers data (as opposed to a `SignatureValue`), 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# Id="PostMarkedData">
      <epm:PostMarkedContent xmlns:epm="http://www.upu.int/EPMService/schemas"
       Encoding="http://www.w3.org/2000/09/xmldsig#base64">RGF0YSBgdG8gcQ...gVGkgMTUgMTI6MTA=</PostMarkedContent>
   </dsig:Object>
```

Example 2 Embedded PostMarkedReceipt:

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 `SignatureSelector` element within the Verify 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 Id="PostMarkedReceiptSignature" xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
      <dsig:SignedInfo>
         <dsig:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
         <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
         <dsig:Reference URI="#TstInfo">
```

```
                <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                <dsig:DigestValue>3Lk/6TE71dqeXZFUJ9qqaPInm24=</dsig:DigestValue>
            </dsig:Reference>
            <dsig:Reference URI="#Receipt">
                <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-xpath-19991116">
                        <dsig:XPath xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
                            ancestor-or-self::dsig:SignatureValue[../@Id!="PostMarkedReceiptSignature"]
                        </dsig:XPath>
                    </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#" Id="TstInfo">
            <dss:TstInfo xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema">
                <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# Id="Receipt">
            <epm:PostMarkedReceipt xmlns:epm="http://www.upu.int/EPMService/schemas">
                <Receipt>
                    <TransactionKey>
                        <Locator>
                            <CountryCode>CA</CountryCode>
                            <Version>114</Version>
                            <ServiceProvider>ePost Corporation</ServiceProvider>
                        <Environment xsi:nil="true"/>
                     </Locator>
                     <Key>1234567890</Key>
                     <Sequence>1</Sequence>
                  </TransactionKey>
                  <Requester>CN=Joe Public, O=VeriSign Class 1 Certificate, C=CA, E=joe.public@rogers.com</Requester>
                  <Operation>Verify</Operation>
                  <TSAX509SubjectName> … </TSAX509SubjectName>
                  <MessageImprint> … </MessageImprint>
                  <PostMarkImage> … </PostMarkImage>
                  <RevocationStatusQualifier>CRL Checked</RevocationStatusQualifier>
                  <TimeStampToken MimeType="application/pkcs7-signature"></TimeStampToken>
                  <ReceiptMetadata>
                     <Name> ... </Name>
                     <Value> … </Value>
                  </ReceiptMetadata>
               </Receipt>
            </epm:PostMarkedReceipt>
         </dsig:Object>
      </dsig:Signature>
<!-- End of PostMarkedReceipt signature -->
<!-- Beginning of signed document being PostMarked -->
   <Object Id="DetachedDataBeingSigned">
      <PersonalData>
         <Name>Ed Smith</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/2001/10/xml-exc-c14n#"/>
         <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 CMS/PKCS7	XMLSIG	Elements Affected	Comments
Input / Request Elements					
OrganizationID	M	✓	✓		Must match the string specified at registration time.
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.
	DocumentContainsTemplate	O		✓	Signatures produced will be returned in `<DocumentWithSignature>`	When users are passing in XMLSig signing templates, they must include this empty element directive.
	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>`.
	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	✓	✓		This element is returned as part of the SignResponse and contains the unique identifier.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. Also initialized when signing templates are used. See also `<SignatureObject>`.	
	PostMarkedReceipt	O	✓	✓	See `<IssuePostMarkedReceipt>` above.	
	SignatureInfo	O	✓	✓	Returned when `<ReturnSignatureInfo>` has been specified.	
	X509Info	O	✓	✓	Returned when `<ReturnX509Info>` has been specified.	

Verify Protocol

	Optionality	As Used in Sig Type CMS/PKCS7	XMLSIG	Elements Affected	Comments
Input / Request Elements					
OrganizationID	M	✓	✓		Must match the string specified at registration time.
SignatureObject	M	✓			Required when verifying CMS/PKCS7 detached signatures.
InputDocuments	O	✓			Default location of “same document” signature to be verified.
InputDocuments	M		✓		Presently constrained to one `<Document>` occurrence. Must contain signature(s) to be verified along with any referenced signed content.
SignatureSelector	O		✓		Optionally used to specify the ancestor node containing the target signature to be verified (NodeName) or the XPath expression to select the signatures to be verified. May apply if more than one signature is present in the InputDocuments.
ClaimedIdentity	O	✓	✓		Optionally used for alternate authentication schemes or when “Proof of Delivery” is required.
Processing Option Flags					
AddTimestamp	O	✓		Updated CMS/PKCS7 signature,	Allows for the inclusion of an RFC3161

					now containing an embedded RFC 3161 timestamp token, is returned in `<SignatureObject>`.	embedded timestamp into the verified CMS/PKCS7 signature.
	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 `SignatureSelector` 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	✓	✓		This element is returned as part of the VerifyResponse and contains the unique identifier.Always initialized and

						returned.
	PostMarkedReceipt	O	✓	✓	See `<IssuePostMarkedReceipt>` above.	
	SignatureObject	O	✓		Initialized for CMS/PKCS7 signatures when `<AddTimestamp>` 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.	

A. Acknowledgements

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

Participants:

Trevor Perrin, individual
Nick Pope, Thales eSecurity
Juan Carlos Cruellas, Centre d'aplicacions avançades d’Internet (UPC)