



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27

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

28	1	Introduction	5
29	1.1	Notation	5
30	1.2	Namespaces	5
31	2	Profile Features	6
32	2.1	Identifier	6
33	2.2	Scope	6
34	2.3	Relationship To Other Profiles	6
35	2.4	Signature Object	6
36	2.5	Transport Binding	6
37	2.6	Security Binding	6
38	2.6.1	Security Requirements	6
39	2.7	Common Elements	6
40	2.7.1	Element <PostMarkedReceipt> and the PostMarkedReceipt Signature	6
41	2.7.2	Input / Output Element <TransactionKey>	8
42	2.7.3	Input Element <OrganizationID>	8
43	2.7.4	Input Element <ContentIdentifier>	9
44	2.7.5	Input / Output Element <ClientApplicationID>	9
45	2.7.6	Input Element <ContentMetaData>	9
46	3	Profile of Signing Protocol	10
47	3.1	Element <SignRequest>	10
48	3.1.1	Constraints on Element <OptionalInputs>	10
49	3.1.1.1	Element SignatureType	10
50	3.1.1.2	Element <KeySelector>	10
51	3.1.1.3	Element <SignedReferences> not Supported	10
52	3.1.1.4	Element <AddTimestamp>	11
53	3.1.1.5	Element <InputDocuments>	12
54	3.1.1.6	Element <SignaturePlacement>	12
55	3.1.2	EPM-specific <OptionalInputs>	13
56	3.1.2.1	Element <DocumentWithTemplate>	13
57	3.1.2.2	Element <TransactionKey>	13
58	3.1.2.3	Element <ClaimedIdentity>	13
59	3.1.2.4	Element <OrganizationID>	14
60	3.1.2.5	Element <ContentIdentifier>	14
61	3.1.2.6	Element <ClientApplicationID>	14
62	3.1.2.7	Element <ContentMetaData>	14
63	3.1.3	<OptionalInputs> Processing Flags	15
64	3.1.3.1	Element <ExtendLifecycle>	15
65	3.1.3.2	Element <EndLifecycle>	15
66	3.1.3.3	Element <IssuePostMarkedReceipt>	15
67	3.1.3.4	Element <StoreNonRepudiationEvidence>	16
68	3.1.3.5	Element <ReturnSignatureInfo>	16
69	3.1.3.6	Element <ReturnX509Info>	16
70	3.2	Element <SignResponse>	16
71	3.2.1	Element <Result>	16
72	3.2.2	Element <SignatureObject>	16

73	3.2.3 EPM-specific <OptionalOutputs>.....	16
74	3.2.3.1 Element <TransactionKey>.....	16
75	3.2.3.2 Element <PostMarkedReceipt>.....	17
76	3.2.3.3 Element <DocumentWithSignature>.....	17
77	3.2.3.4 Element <SignatureInfo>.....	17
78	3.2.3.5 Element <X509Info>.....	17
79	3.2.4 Timestamp Handling Profile of Sign Protocol.....	18
80	3.2.4.1 Standalone Timestamp from <DocumentHash>.....	18
81	3.2.4.2 Standalone Timestamp from <Document>.....	18
82	3.2.4.3 Embedding a Signature Timestamp into a user-provided Signature.....	18
83	4 Profile of Verifying Protocol.....	20
84	4.1 Element <VerifyRequest>.....	20
85	4.1.1 Constraints on Element <OptionalInputs>.....	20
86	4.1.1.1 Element <SignatureObject>.....	20
87	4.1.1.2 Element <InputDocuments>.....	20
88	4.1.1.3 Element <VerifyManifests>.....	20
89	4.1.1.4 Element <VerificationTime>.....	20
90	4.1.1.5 Element <AdditionalKeyInfo>.....	20
91	4.1.1.6 Element <ReturnProcessingDetails>.....	20
92	4.1.1.7 Element <ReturnSigningTime>.....	20
93	4.1.1.8 Element <ReturnSignerIdentity>.....	20
94	4.1.1.9 Element <ReturnUpdatedSignature>.....	20
95	4.1.1.10 Element <ReturnTransformedDocument>.....	20
96	4.1.2 EPM-specific <OptionalInputs>.....	21
97	4.1.2.1 Element <TransactionKey>.....	21
98	4.1.2.2 Element <OrganizationID>.....	21
99	4.1.2.3 Element <ContentIdentifier>.....	21
100	4.1.2.4 Element <ClientApplicationID>.....	21
101	4.1.2.5 Element <ContentMetaData>.....	21
102	4.1.3 <OptionalInputs> Processing Flags.....	21
103	4.1.3.1 Element <ExtendLifecycle>.....	21
104	4.1.3.2 Element <EndLifecycle>.....	21
105	4.1.3.3 Element NodeName.....	21
106	4.1.3.4 Element <IssuePostMarkedReceipt>.....	22
107	4.1.3.5 Element <StoreNonRepudiationEvidence>.....	22
108	4.1.3.6 Element <ReturnSignatureInfo>.....	22
109	4.1.3.7 Element <ReturnX509Info>.....	22
110	4.2 Element <VerifyResponse>.....	22
111	4.2.1 Element <Result>.....	22
112	4.2.2 Element <SignatureObject>.....	23
113	4.2.3 Element <OptionalOutputs>.....	23
114	4.2.3.1 Element <DocumentWithSignature>.....	23
115	4.2.4 Element <EPM-specific OptionalOutputs>.....	23
116	4.2.4.1 Element <TransactionKey>.....	23
117	4.2.4.2 Element <PostMarkedReceipt>.....	23
118	4.2.4.3 Element <SignatureInfo>.....	23
119	4.2.4.4 Element <X509Info>.....	23
120	5 Signing Template Examples.....	24
121	6 PostMarkedReceipt Examples.....	28
122	7 Element cross-reference Table.....	33

123	8	References	40
124		8.1 Normative	40
125		Appendix A. Revision History.....	41
126		Appendix B. Notices.....	42
127			

128 1 Introduction

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

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

136 1.1 Notation

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

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

144 1.2 Namespaces

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

148 This schema is associated with the following XML namespace:

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

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

151 Conventional XML namespace prefixes are used in this document:

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

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

160 2 Profile Features

161 2.1 Identifier

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

163 2.2 Scope

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

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

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

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

179 2.3 Relationship To Other Profiles

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

181 2.4 Signature Object

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

183 2.5 Transport Binding

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

186 2.6 Security Binding

187 2.6.1 Security Requirements

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

189 2.7 Common Elements

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

191 2.7.1 Element <PostMarkedReceipt> and the PostMarkedReceipt Signature

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

199 ➤ **Sign**

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

202 ➤ **Verify**

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

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

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

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

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

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

221 ➤ <dsig:Reference> whose URI attribute references a <dsig:Object> containing the <TstInfo>

222 ➤ <dsig:Reference> whose URI attribute references a <dsig:Object> containing the
223 <epm:PostMarkedReceipt>

224 ➤ <dsig:Reference> whose URI attribute references a <dsig:Object> containing the
225 <dsig:SignatureValue> of the signature being PostMarked (Sign) or Verified and PostMarked
226 (Verify)

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

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

229

```
230 <xs:element ref="epm:PostMarkedReceipt "  
231  
232 <!-- imported from the EPM schema -->  
233 <xs:element name="PostMarkedReceipt" type="epm:PostMarkedReceiptType">  
234  
235 <xs:complexType name="PostMarkedReceiptType">  
236 <xs:sequence>  
237 <xs:element name="Receipt" type="epm:ReceiptType"/>  
238 <xs:element name="TimeStampToken" type="epm:QualifiedDataType" minOccurs="0"/>  
239 <xs:element name="ReceiptSignature" type="epm:QualifiedDataType" minOccurs="0"/>  
240 </xs:sequence>  
241 </xs:complexType>  
242
```

```

243 <xs:complexType name="ReceiptType">
244   <xs:sequence>
245     <xs:element name="TransactionKey" type="epm:TransactionKeyType"/>
246     <xs:element name="TSAX509SubjectName" type="xs:string"/>
247     <xs:element name="TimeStampValue" type="xs:string"/>
248     <xs:element name="RevocationStatusQualifier" type="xs:string"/>
249     <xs:element name="ClientApplication" type="epm:ClientApplicationType"/>
250     <xs:element name="ResourceID" type="xs:string"/>
251   </xs:sequence>
252 </xs:complexType>
253

```

254

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

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

263 2.7.2 Input / Output Element <TransactionKey>

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

273

```

274 <xs:element ref="epm:TransactionKey"
275
276 <!-- imported from the EPM schema -->
277 <xs:element name="TransactionKey" type="epm:TransactionKeyType">
278 <xs:complexType name="TransactionKeyType">
279   <xs:sequence>
280     <xs:element name="Locator" type="epm:LocatorType"/>
281     <xs:element name="Key" type="xs:string"/>
282     <xs:element name="Sequence" type="xs:string"/>
283   </xs:sequence>
284 </xs:complexType>
285 <xs:complexType name="LocatorType">
286   <xs:sequence>
287     <xs:element name="CountryCode" type="xs:string"/>
288     <xs:element name="Version" type="xs:string"/>
289     <xs:element name="ServiceProvider" type="xs:string" nillable="true"/>
290     <xs:element name="Environment" type="xs:string" nillable="true"/>
291   </xs:sequence>
292 </xs:complexType>

```

293

294 2.7.3 Input Element <OrganizationID>

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

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

299

300 **2.7.4 Input Element <ContentIdentifier>**

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

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

305

306 **2.7.5 Input / Output Element <ClientApplicationID>**

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

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

311

312 **2.7.6 Input Element <ContentMetaData>**

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

318 3 Profile of Signing Protocol

319 3.1 Element <SignRequest>

320 3.1.1 Constraints on Element <OptionalInputs>

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

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

325 3.1.1.1 Element SignatureType

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

327 The following <SignatureType> URNs are supported:

328 ➤ Signature creation URNs:

- 329 ▪ urn:ietf:rfc:3275 (i.e. an XML Digital Signature)
- 330 ▪ urn:ietf:rfc:3369 (i.e. a CMS/PKCS7 binary Signature)

331 ➤ Timestamp creation URNs:

- 332 ▪ oasis:names:tc:dss:1.0:core:schema:XMLTimeStampToken
- 333 ▪ urn:ietf:rfc:3161 (i.e. a CMS timestamp token)

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

341 3.1.1.2 Element <KeySelector>

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

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

351 3.1.1.3 Element <SignedReferences> not Supported

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

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

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

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

369 3.1.1.4 Element <AddTimestamp>

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

377 ➤ For <SignatureType> values of urn:ietf:rfc:3369 (i.e. CMS/PKCS7), the timestamp will be produced as
378 follows:

379 ▪ The following object identifier identifies the Signature Timestamp attribute:

380 ▪ id-aa-signatureTimeStampToken OBJECT IDENTIFIER ::=

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

382 i.e. 1.2.840.113549.1.9.16.2.14

383 ▪ The value of messageImprint field within TimeStampToken shall be a hash of the value of
384 signature field within SignerInfo for the signedData being timestamped.

385 ▪ The RFC 3161 timestamp token will be added as an unauthenticated attribute, as defined above,
386 of the generated signature. The result will be returned in <SignatureObject>.

387 ▪ This approach is identical to that prescribed by ETSI TS 101 733 and is the convention utilized
388 and supported by the EPM profile.

389 ➤ For <SignatureType> values of urn:ietf:rfc:3275 (i.e. [XMLSig]), the timestamp will be produced as
390 follows:

391 ▪ the timestamp will be produced as defined in section 5.1 of [DSSCore].

392 ▪ The URI attribute of the first <Reference> element of the timestamp's <SignedInfo> will point
393 to an <Object> containing the <TstInfo> element.

394 ▪ The URI attribute of the second <Reference> element of the timestamp's <SignedInfo> will
395 point to an <Object> containing the <SignatureValue> element of the signature just created.
396 This is because the timestamp being added is a signature timestamp and not a content
397 timestamp.

398 ▪ The detached timestamp signature will be included as the first child of the XML document root
399 and will immediately precede the generated signature.

400 ➤ The <AddTimestamp> optional input is not supported on timestamp-related <SignatureType>'s as
401 used within the Signing protocol, since one cannot add a timestamp to a timestamp.

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

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

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

412 3.1.1.5 Element <InputDocuments>

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

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

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

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

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

432

433 3.1.1.6 Element <SignaturePlacement>

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

437 ➤ Enveloped Signatures (XMLSig)

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

442 ➤ Enveloping Signatures (XMLSig)

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

- 449 ➤ Enveloping Signatures (CMS/PKCS7)
- 450 In order to obtain an enveloping CMS/PKCS7 signature, the caller must include the
- 451 `<EnvelopingSignature>` optional input. When specified, the input document's content will be
- 452 encapsulated in the CMS/PKCS7 SignedData signature structure and returned in the
- 453 `<SignatureObject>` element. When the `<EnvelopingSignature>` optional input is specified, **only**
- 454 **one** `<InputDocuments>` occurrence is supported (i.e. can only have a single child), and the
- 455 `WhichDocument` attribute, if present, will be ignored.
- 456 ➤ Detached Signatures (CMS/PKCS7)
- 457 This is default processing for CMS/PKCS7 signatures and is identical to **[DSSCore]**. The input
- 458 document's content will **NOT** be encapsulated in the CMS/PKCS7 SignedData signature structure as per
- 459 detached signature conventions. This signature will be returned in the `<SignatureObject>` element.
- 460 ➤ Same Document Detached Signatures (XMLSig)
- 461 Callers can obtain detached XMLSig signatures by constructing a "signing template" reflecting a same
- 462 document detached signature in the `<DocumentWithTemplate>` input element. A same document
- 463 detached XMLSig signature will also be produced under an `<IssuePostMarkedReceipt>` scenario
- 464 when the `Location` attribute is specified as embedded. See section 3.1.3.3. These signatures are
- 465 returned in the `<DocumentWithSignature>` element.
- 466 This optional input does not apply when users are requesting a timestamp `<SignatureType>` in which case
- 467 placement is determined by the server.
- 468 When greater and more direct control of signature placement is required, users must use an EPM signing
- 469 template. EPM signing templates should be used if user wishes to have a same-document detached XMLSig-
- 470 based signature created. See section 3.1.2.1 for details.
- 471 **Note:** As with **[DSSCore]**, **[XMLSig]** signatures created from `<DocumentHash>` will be returned in
- 472 `<SignatureObject>`.

473 3.1.2 EPM-specific `<OptionalInputs>`

474 The following additional elements are specific to the EPM profile. Their specific usage and constraints are

475 documented below.

476 3.1.2.1 Element `<DocumentWithTemplate>`

477 The `<DocumentWithTemplate>` optional input element is used when users elect to utilize the EPM's "signing

478 template" mechanism. EPM-supported signing templates contain not only the data to be signed, but also the

479 format and directives of the signature to be created, expressed as valid **[XMLSig]** elements. In this fashion more

480 elaborate signatures involving transforms, signed and unsigned properties, manifests, and multiple

481 `<Reference>` elements can be supported. **[XMLSig]** elements such as `<SignatureValue>`,

482 `<DigestValue>`, and `<X509Certificate>` are populated by the EPM service. The user simply leaves the

483 element tags empty, and the EPM service will automatically include the generated content and return the signed

484 document in the `<DocumentWithSignature>` element of the `<SignResponse>`. See Example 1 in section 5

485 for an example of signing templates. More details are available in the EPM Systems Integrator's Guide and other

486 EPM documentation available at the UPU's Postal Technology Centre <http://www.ptc.upu.int/>.

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

488

489 3.1.2.2 Element `<TransactionKey>`

490 Please refer to the description in section 2.7.2 entitled [Element `<TransactionKey>`](#)

491 3.1.2.3 Element `<ClaimedIdentity>`

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

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

509

```
510 <xs:element name="ClaimedIdentity">  
511   <xs:complexType>  
512     <xs:sequence>  
513       <xs:element name="Name" type="saml:NameIdentifierType"/>  
514       <xs:element ref="epm:SupportingInfo"/>  
515     </xs:sequence>  
516   </xs:complexType>  
517 </xs:element>  
  
519 <!-- imported from the EPM schema -->  
520 <xs:element name="SupportingInfo" type="epm:SupportingInfoType"/>  
521 <xs:complexType name="SupportingInfoType">  
522   <xs:element name="BasicAuth" type="epm:BasicAuthType" nillable="true"/>  
523   <xs:element name="RequesterSignature" type="epm:QualifiedDataType" nillable="true"/>  
524   <xs:element name="AlternateIdentity" type="epm:AlternateIdentityType" nillable="true"/>  
525 </xs:complexType>  
526  
527 <xs:complexType name="epm:QualifiedDataType">  
528   <xs:simpleContent>  
529     <xs:extension base="xs:base64Binary">  
530       <xs:attribute name="MimeType" type="xs:string"/>  
531     </xs:extension>  
532   </xs:simpleContent>  
533 </xs:complexType>
```

534

535 **3.1.2.4 Element <OrganizationID>**

536 Please refer to the description in section 2.7.3 entitled [Input Element <OrganizationID>](#)

537 **3.1.2.5 Element <ContentIdentifier>**

538 Please refer to the description in section 2.7.4 entitled [Input Element <ContentIdentifier>](#)

539 **3.1.2.6 Element <ClientApplicationID>**

540 Please refer to the description in section 2.7.5 entitled [Input / Output Element <ClientApplicationID>](#)

541 **3.1.2.7 Element <ContentMetaData>**

542 Please refer to the description in section 2.7.6 entitled [Input Element <ContentMetaData>](#)

543 3.1.3 <OptionalInputs> Processing Flags

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

547 3.1.3.1 Element <ExtendLifecycle>

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

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

552 3.1.3.2 Element <EndLifecycle>

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

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

556

557 3.1.3.3 Element <IssuePostMarkedReceipt>

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

- 564 ➤ For a Location attribute value of standalone regardless of the <SignatureType>, processing is as
565 follows:
 - 566 ▪ The <PostMarkedReceipt> XML element will be returned as a standalone optional output
567 structure as defined in section 2.7.1. Standalone <PostMarkedReceipt>'s are self-contained
568 and contain a timestamp signature which binds the receipt to the signature value of the signature
569 being created as part of this Sign operation.
- 570 ➤ For a Location attribute value of embedded and a <SignatureType> value of urn:ietf:rfc:3275 (i.e.
571 XMLSig), processing is as follows:
 - 572 ▪ The EPM will first create an [XMLSig] based "detached" signature covering the input document.
573 The input document's contents will be outside the produced signature and referenced by it. The
574 EPM will then add a <PostMarkedReceipt> detached signature structure covering the
575 <SignatureValue> of the first signature just created. The resulting signed and PostMarked
576 document will be returned in the <DocumentWithSignature> element.
- 577 ➤ A Location attribute value of embedded with a <SignatureType> value of urn:ietf:rfc:3369 (i.e.
578 CMS/PKCS7) is not supported.
 - 579 ▪ A signature timestamp (i.e. an RFC 3161 timestamp token) however can be embedded in a
580 CMS/PKCS7 signature by using the <AddTimestamp> optional input described in section
581 3.1.1.4. This timestamp bears the Issuer name of the Post's TimeStamp Authority.

582

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

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

588 `</xs:element>`

589 **3.1.3.4 Element <StoreNonRepudiationEvidence>**

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

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

593

594 **3.1.3.5 Element <ReturnSignatureInfo>**

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

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

599

600 **3.1.3.6 Element <ReturnX509Info>**

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

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

605

606 **3.2 Element <SignResponse>**

607 **3.2.1 Element <Result>**

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

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

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

611 **3.2.2 Element <SignatureObject>**

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

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

619 **3.2.3 EPM-specific <OptionalOutputs>**

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

622 **3.2.3.1 Element <TransactionKey>**

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

625

626 **3.2.3.2 Element <PostMarkedReceipt>**

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

629

630 **3.2.3.3 Element <DocumentWithSignature>**

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

635

636 **3.2.3.4 Element <SignatureInfo>**

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

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

```
647 <xs:element ref="epm:SignatureInfo"
648
649 <!-- imported from the EPM schema -->
650 <xs:element name="SignatureInfo" type="epm:SignatureInfoType">
651 <xs:complexType name="SignatureInfoType">
652 <xs:sequence>
653 <xs:element name="SignedContent" type="base64Binary" nillable="true"/>
654 <xs:element name="ContentHash" type="xs:string"/>
655 <xs:element name="ContentHashAlgo" type="xs:string"/>
656 <xs:element name="ContentEncryptAlgo" type="xs:string"/>
657 <xs:element name="SigningTime" type="xs:string"/>
658 <xs:element name="PKCS1" type="epm:QualifiedDataType" nillable="true"/>
659 </xs:sequence>
660 </xs:complexType>
```

661

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

663 **3.2.3.5 Element <X509Info>**

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

672

```

673 <xs:element ref="epm:X509Info"
674
675 <!-- imported from the EPM schema -->
676 <xs:element name="X509Info" type="epm:X509InfoType">
677 <xs:complexType name="X509InfoType">
678   <xs:sequence>
679     <xs:element name="X509Subject" type="xs:string"/>
680     <xs:element name="X509Issuer" type="xs:string"/>
681     <xs:element name="X509Serial" type="xs:string"/>
682     <xs:element name="X509StatusSource" type="xs:string"/>
683     <xs:element name="X509ValidFrom" type="xs:string"/>
684     <xs:element name="X509ValidTo" type="xs:string"/>
685     <xs:element name="X509Certificate" type="xs:string"/>
686     <xs:element name="X509RevocationReason" type="xs:string" nillable="true"/>
687     <xs:element name="X509RevocationReason" type="xs:string" nillable="true"/>
688     <xs:element name="X509RevocationTime" type="xs:string" nillable="true"/>
689     <xs:element name="X509ValidationData" type="xs:base64Binary" nillable="true"/>
690   </xs:sequence>
691 </xs:complexType>
692

```

693

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

695 3.2.4 Timestamp Handling Profile of Sign Protocol

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

703 3.2.4.1 Standalone Timestamp from `<DocumentHash>`

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

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

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

- 709 ➤ `<dss:Timestamp>` of type `RFC3161TimeStampToken`
- 710 ➤ `<dss:Timestamp>` of type `<dsig:Signature>` formatted as an XML Timestamp Token as per section
711 5.1.1 of **[DSSCore]**

712 3.2.4.2 Standalone Timestamp from `<Document>`

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

718 3.2.4.3 Embedding a Signature Timestamp into a user-provided Signature

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

- 722 ➤ For <SignatureType> values of urn:ietf:rfc:3161
- 723 ▪ The EPM detects from the MimeType attribute (of the <Base64Data> element of the
- 724 <Document> element) that the input is a CMS/PKCS7 signature. The EPM will embed an RFC
- 725 3161 timestamp token into the incoming signature and return the resultant CMS/PKCS7 in the
- 726 <SignatureObject> element. The timestamp will be embedded as an unsigned attribute as
- 727 specified in section 3.1.1.4
- 728 ➤ For <SignatureType> values of oasis:names:tc:dss:1.0:core:schema:XMLTimeStampToken
- 729 ▪ Users must pass in an XMLSig compliant signature in the <Document> input element. There
- 730 must be at most a single signature in the <Document> element and the signature value within
- 731 that signature must be specified as <dsig:SignatureValue>. The EPM will return a
- 732 <dss:timestamp> as a standalone <dsig:Signature> in the <SignatureObject>
- 733 element. The <dss:timestamp> will be very similar to Example 1 in section 6 except that the
- 734 2nd <Reference> will not be present. The user must splice the returned <dss:timestamp>
- 735 into the signed document as desired. It should be noted that timestamping in this manner works
- 736 best when timestamping detached signatures so as not to invalidate the original signature after
- 737 splicing the timestamp signature back into the original document.
- 738 It should be noted that no verification is performed on the incoming signature prior to timestamping. A signature
- 739 timestamp calculated over the signature value of the incoming signature and the subsequent embedding of that
- 740 timestamp into the incoming signature is all that is performed. For timestamping of verified signatures, please
- 741 refer to the Verify Protocol below.

742

743 4 Profile of Verifying Protocol

744 4.1 Element <VerifyRequest>

745 4.1.1 Constraints on Element <OptionalInputs>

746 4.1.1.1 Element <SignatureObject>

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

749 4.1.1.2 Element <InputDocuments>

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

753 4.1.1.3 Element <VerifyManifests>

754 This optional input element is not supported by the EPM.

755 4.1.1.4 Element <VerificationTime>

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

758 4.1.1.5 Element <AdditionalKeyInfo>

759 This optional input element is not required by the EPM.

760 4.1.1.6 Element <ReturnProcessingDetails>

761 This optional input element is not supported by the EPM.

762 4.1.1.7 Element <ReturnSigningTime>

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

766 4.1.1.8 Element <ReturnSignerIdentity>

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

769 4.1.1.9 Element <ReturnUpdatedSignature>

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

776 4.1.1.10 Element <ReturnTransformedDocument>

777 This optional input element is not supported by the EPM.

778 4.1.2 EPM-specific <OptionalInputs>

779 4.1.2.1 Element <TransactionKey>

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

781 4.1.2.2 Element <OrganizationID>

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

783 4.1.2.3 Element <ContentIdentifier>

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

785 4.1.2.4 Element <ClientApplicationID>

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

787 4.1.2.5 Element <ContentMetaData>

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

789 4.1.3 <OptionalInputs> Processing Flags

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

793 4.1.3.1 Element <ExtendLifecycle>

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

795 4.1.3.2 Element <EndLifecycle>

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

797 4.1.3.3 Element NodeName

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

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

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

```
806 <lgl:Party1>  
807   <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">  
808     ...  
809   </dsig:Signature>  
810 </lgl:Party1>  
811 <lgl:Party2>  
812   <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">  
813     ...  
814   </dsig:Signature>  
815 </lgl:Party2>
```

816

817

818 4.1.3.4 Element <IssuePostMarkedReceipt>

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

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

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

825 ➤ For a Location attribute value of standalone regardless of the <SignatureType>, processing is as
826 follows:

827 ▪ The <PostMarkedReceipt> XML element will be returned as a standalone optional output
828 structure as defined in section 2.7.1. Standalone <PostMarkedReceipt>'s are self-contained
829 and contain a timestamp signature which binds the receipt to the signature value of the signature
830 being verified as part of this Verify operation.

831 ➤ For a Location attribute value of embedded and a <SignatureType> value of urn:ietf:rfc:3275 (i.e.
832 XMLSig), the incoming <Document> containing the signature(s) **must** be a **detached** XMLSig based
833 signature. Processing is as follows:

834 The incoming signed document will contain an **[XMLSig]** based “detached” signature covering the
835 required content within the input document. The input document’s signed content will be outside the
836 signature and referenced by it. The EPM will verify this signature. If the signature(s) verify successfully,
837 the EPM will then add a <PostMarkedReceipt> detached signature structure covering the
838 <SignatureValue>'s of the signature(s) just verified.

839 ▪ The resulting PostMarked document will be returned in the <DocumentWithSignature>
840 element and will include the <PostMarkedReceipt> attesting to its validity.

841 ➤ A Location attribute value of embedded with a <SignatureType> value of urn:ietf:rfc:3369 (i.e.
842 CMS/PKCS7) is not supported.

843 ▪ A signature timestamp (i.e. an RFC 3161 timestamp token) however can be embedded in a
844 CMS/PKCS7 signature by using the <AddTimestamp> optional input described in section
845 3.1.1.4. This timestamp bears the Issuer name of the Post’s TimeStamp Authority.

846

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

```
848 <xs:element name="IssuePostMarkedReceipt">  
849   <xs:complexType>  
850     <xs:attribute name="Location" type="xs:string"/>  
851   </xs:complexType>  
852 </xs:element>
```

853 4.1.3.5 Element <StoreNonRepudiationEvidence>

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

855 4.1.3.6 Element <ReturnSignatureInfo>

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

857 4.1.3.7 Element <ReturnX509Info>

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

859 4.2 Element <VerifyResponse>

860 4.2.1 Element <Result>

861 This profile defines an additional <ResultMajor> code as follows:
862 urn:oasis:names:tc:dss:1.0:resultmajor:Warning
863 All EPM result codes are always accompanied by a <ResultMessage> element.

864 **4.2.2 Element <SignatureObject>**

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

867 **4.2.3 Element <OptionalOutputs>**

868 **4.2.3.1 Element <DocumentWithSignature>**

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

872 **4.2.4 Element <EPM-specific OptionalOutputs>**

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

875 **4.2.4.1 Element <TransactionKey>**

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

878 **4.2.4.2 Element <PostMarkedReceipt>**

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

882 **4.2.4.3 Element <SignatureInfo>**

883 See section 0 for a detailed explanation of this element's usage.

884 **4.2.4.4 Element <X509Info>**

885 See section 3.2.3.5 for a detailed explanation of this element's usage.

886

5 Signing Template Examples

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

889

890

Example 1:

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

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

918

919

920

921

922

923

924

925

926

927

```

<?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>

```



```
928         </X509Data>
929     </KeyInfo>
930 </Signature>
931 </DocumentWithTemplate>
```

932

933 **Example 2:**

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

```
936
937 <?xml version="1.0" encoding="UTF-8"?>
938 <DocumentWithTemplate>
939     <Data>
940         <SubData1>
941             <SubSubData1 MimeType="text/plain">This is the data to be signed.</SubSubData1>
942             <SubSubData2 MimeType="text/plain">This is the data to be signed.</SubSubData2>
943             <SubSubData3 MimeType="text/plain">This is the data to be signed.</SubSubData3>
944         </SubData1>
945         <SubData2>This is data.</SubData2>
946         <SubData3>This is data.</SubData3>
947     </Data>
948     <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
949         <SignedInfo>
950             <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
951             <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
952                 <Reference URI="#xpointer(/DocumentWithTemplate/Data/SubData1)">
953                     <Transforms>
954                         <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
955                     </Transforms>
956                     <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
957                         <DigestValue></DigestValue>
958                     </Reference>
959                 </SignedInfo>
960                 <SignatureValue>
961                 </SignatureValue>
962                 <KeyInfo>
963                     <X509Data>
964                         <X509Certificate></X509Certificate>
965                         <X509SubjectName></X509SubjectName>
966                         <X509IssuerSerial>
967                             </X509IssuerSerial>
968                     </X509Data>
969                 </KeyInfo>
970             </Signature>
971 </DocumentWithTemplate>
```

972

973 **Example 3:**

974 This is a more complicated example using intersect and subtract XPath Filters. This 3rd example illustrates step 1 in a multi-party contract signing
975 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
976 first party has signed the document. This second template would simply change the "subtract" value in the transform filter. Again certificate
977 information is added to the produced signature.

```
978 <?xml version="1.0"?>
979 <DocumentWithTemplate>
980   <Contract>
981     <Terms MimeType="text/plain">This is the data to be signed by both parties</Terms>
982     <Conditions MimeType="text/plain">This is the data to be signed by both parties</Conditions>
983     <Obligations MimeType="text/plain">This is the data to be signed by both parties</Obligations>
984     <Party1>
985       <Terms MimeType="text/plain">This is the data to be signed by party 1</Terms>
986       <Conditions MimeType="text/plain">This is the data to be signed by party 1</Conditions>
987       <Obligations MimeType="text/plain">This is the data to be signed by party 1</Obligations>
988     </Party1>
989     <Party2>
990       <Terms MimeType="text/plain">This is the data to be signed by party 2</Terms>
991       <Conditions MimeType="text/plain">This is the data to be signed by party 2</Conditions>
992       <Obligations MimeType="text/plain">This is the data to be signed by party 2</Obligations>
993     </Party2>
994   </Contract>
995   <dsig:Signature xmlns:dsig-xpath="http://www.w3.org/2002/06/xmldsig-filter2">
996     <dsig:SignedInfo>
997       <dsig:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
998       <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
999       <dsig:Reference URI="">
1000         <dsig:Transforms>
1001           <dsig:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
1002           <dsig:Transform Algorithm="http://www.w3.org/2002/06/xmldsig-filter2">
1003             <dsig-xpath:XPath Filter="intersect"//Contract</dsig-xpath:XPath>
1004             <dsig-xpath:XPath Filter="subtract"//Party2</dsig-xpath:XPath>
1005           </dsig:Transform>
1006         </dsig:Transforms>
1007       <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1008       <dsig:DigestValue></dsig:DigestValue>
1009     </dsig:Reference>
1010   </dsig:SignedInfo>
1011   <dsig:SignatureValue>
1012 </dsig:SignatureValue>
1013 <dsig:KeyInfo>
1014   <dsig:X509Data>
1015     <dsig:X509Certificate></dsig:X509Certificate>
1016     <dsig:X509SubjectName></dsig:X509SubjectName>
1017     <dsig:X509IssuerSerial></dsig:X509IssuerSerial>
1018   </dsig:X509Data>
1019 </dsig:Signature>
```

```
1020     </dsig:KeyInfo>
1021     </dsig:Signature>
1022 </DocumentWithTemplate>
1023
```

6 PostMarkedReceipt Examples

1024

1025 PostMarked receipts are normally returned to the application as standalone XML structures, whether they are of type CMS/PKCS7 or XMLSig.
1026 Upon request however <PostMarkedReceipt>'s can be embedded in the incoming signed document. This is true for both the Sign protocol as
1027 well as the Verify protocol. The first example below is a standalone <PostMarkedReceipt>, and the second example is one that is embedded
1028 into the signed document.

1029 Example 1:

1030 This is an example of a PostMarkedReceipt. It is essentially a conventional XMLSig enveloping signature over the <SignatureValue> of the
1031 target signature being PostMarked. It contains three (3) <Reference> elements pointing to each of the following:

- 1032 ➤ a standard <dss:TstInfo> as per [DSSCore]
- 1033 ➤ an <epm:PostMarkedReceipt> element from the [EPM] schema
- 1034 ➤ the <SignatureValue> element of the target signature being PostMarked

1035 Selected element contents have been deliberately truncated for brevity and clarity.

1036

1037 <?xml version="1.0" encoding="UTF-8"?>

1038 <dsig:Signature Id="PostMarkedReceipt_001" xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">

1039 <dsig:SignedInfo>

1040 <dsig:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>

1041 <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>

1042 <dsig:Reference URI="#TstInfo040327174718Z">

1043 <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

1044 <dsig:DigestValue>jWkUFR6epvkrtaxTiQ33DiWy+l8=</dsig:DigestValue>

1045 </dsig:Reference>

1046 <dsig:Reference URI="#Receipt040327174718Z">

1047 <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

1048 <dsig:DigestValue>9JWkdLh/8Cs9Slu2QmZixOJl+x0=</dsig:DigestValue>

1049 </dsig:Reference>

1050 <dsig:Reference URI="#PostMarkedSignature">

1051 <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

1052 <dsig:DigestValue>MOBYPfrllMBcJz6yobjhrwH9KP4=</dsig:DigestValue>

1053 </dsig:Reference>

1054 </dsig:SignedInfo>

1055 <dsig:SignatureValue>qnbVJoSgo40oiYyaE3AwBL5/EDq7BhTT6 ... QwllHK+zxy66I=</dsig:SignatureValue>

1056 <dsig:KeyInfo>

1057 <dsig:KeyName>C=CA, O=CPC, OU=EPM Service, CN=EPM Signature, E= ... </dsig:KeyName>

1058 <dsig:X509Data>

1059 <X509Certificate xmlns="http://www.w3.org/2000/09/xmldsig#">MIIEUDC ... EwZOBg==</X509Certificate>

1060 <X509SubjectName xmlns="http:// ... xmldsig#"> C=CA, O=CPC, OU=EPM Service, CN=EPM Signature, E=... </X509SubjectName>

1061 <X509IssuerSerial xmlns="http://www.w3.org/2000/09/xmldsig#">

1062 <X509IssuerName>C=CA, O=CPC, OU=EPM Service, CN=Electronic PostMark CA, E=... </X509IssuerName>

```

1063         <X509SerialNumber>25</X509SerialNumber>
1064     </X509IssuerSerial>
1065 </dsig:X509Data>
1066 </dsig:KeyInfo>
1067 <dsig:Object xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
1068     <dss:TstInfo xmlns:dss="http://www.docs.oasis-open.org/dss/2004/06/oasis-dss-1.0-core-schema-wd-30.xsd"
1069         Id="TstInfo040327174718Z">
1070         <SerialNumber>1847365279</SerialNumber>
1071         <CreationTime>2004-03-27T17:47:18.750</CreationTime>
1072         <Policy/>
1073         <ErrorBound/>
1074         <Ordered/>
1075         <TSA>C=CA, O=CPC, OU=EPM Service, CN=EPM Signature, E= ... </TSA>
1076     </dss:TstInfo>
1077 </dsig:Object>
1078 <dsig:Object xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
1079     <epm:PostMarkedReceipt xmlns:epm="http://www.upu.int/EPMSERVICE" Id="Receipt040327174718Z">
1080         <Receipt>
1081             <TransactionKey>
1082                 <Locator>CA</Locator>
1083                 <Key>1234567890</Key>
1084                 <Sequence>1</Sequence>
1085             </TransactionKey>
1086             <ServiceProviderURL>http://epmproxy.avalonworks.com/upubeta/</ServiceProviderURL>
1087             <TSAX509SubjectName>C=CA, O=CPC, OU=EPM Service, CN=EPM Signature, ... </TSAX509SubjectName>
1088             <TimeStampValue>040327174718Z</TimeStampValue>
1089             <RevocationStatusQualifier>CRL Checked</RevocationStatusQualifier>
1090             <ResourceID>12345678</ResourceID>
1091         </Receipt>
1092         <TimeStampToken MimeType="application/pkcs7-signature"/>
1093     </epm:PostMarkedReceipt>
1094 </dsig:Object>
1095 <dsig:Object xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
1096     <epm:PostMarkedSignatureValue xmlns:epm="http://www.upu.int/EPMSERVICE"
1097         Id="PostMarkedSignature">Vif2Y7aShziFOCy0sDUOR9XVnVCy8LW9hY ... 2RsmQETPmsM=</epm:PostMarkedSignatureValue>
1098 </dsig:Object>
1099 </dsig:Signature>
1100
1101
1102
1103
1104
1105
1106

```

1107 **Note:** Similarly, when the <PostMarkedReceipt>'s signature scope simply covers data, as when used as described in section 3.2.4.2, then the
1108 3rd <Reference> will be to an <Object> containing the hash of the data to be PostMarked with base64 encoding specified.
1109 <dsig:Object xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">

```
1110     <PostMarkedData Id="PostMarked"
1111     Encoding="http://www.w3.org/2000/09/xmldsig#base64">RGF0YSBJIHdhbnQgdG8gcQ...gVGvzdCBNYXkgMTUgMTI6MTA=</PostMarkedData>
1112 </dsig:Object>
```

1113

1114 **Example 2:**

1115 This is an example of an embedded <PostMarkedReceipt> returned after a successful Verify operation. It is a conventional XMLSig detached
1116 signature over the <SignatureValue> of the target signature(s) being PostMarked. It contains three (3) <Reference> elements pointing to
1117 each of the following:

- 1118 ➤ a standard <dss:TstInfo> as per [DSSCore]
- 1119 ➤ an <epm:PostMarkedReceipt> element from the [EPM] schema
- 1120 ➤ the <SignatureValue> element of the target signature(s) being PostMarked

1121 Note that depending on the value of the optional NodeName element specified on the <IssuePostMarkedReceipt> within the request, the
1122 <PostMarkedReceipt> can potentially cover all <SignatureValue>'s in the signed document when the document contains multiple
1123 signatures.

1124 Selected element contents have been deliberately truncated for brevity and clarity.

```
1125
1126 <?xml version="1.0" encoding="UTF-8"?>
1127 <!DOCTYPE Document [
1128 <!ATTLIST Object Id ID #IMPLIED>
1129 ]>
1130 <Document>
1131 <!-- Beginning of PostMarkedReceipt signature -->
1132     <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
1133         <dsig:SignedInfo>
1134             <dsig:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
1135             <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
1136             <dsig:Reference URI="#TstInfo040327174718Z">
1137                 <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1138                 <dsig:DigestValue>3Lk/6TE7ldqeXZFUJ9qqaPInm24=</dsig:DigestValue>
1139             </dsig:Reference>
1140             <dsig:Reference URI="#Receipt040327174718Z">
1141                 <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1142                 <dsig:DigestValue>430zTvcoa9r8Rpr5DiVZf7IPvl8=</dsig:DigestValue>
1143             </dsig:Reference>
1144             <dsig:Reference URI="">
1145                 <dsig:Transforms>
1146                     <dsig:Transform Algorithm="http://www.w3.org/TR/1999/REC-xslt-19991116">
1147                         <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
1148                             xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" version="1.0">
1149                             <xsl:template match="/">
1150                                 <xsl:copy>
```

```

1151         <xsl:copy-of select="//dsig:Signature[position() != 1]/dsig:SignatureValue" />
1152     </xsl:copy>
1153 </xsl:template>
1154 </xsl:stylesheet>
1155 </dsig:Transform>
1156 </dsig:Transforms>
1157 <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1158 <dsig:DigestValue>LRAX6mCfAq8hprb8UMU1H35PTYw=</dsig:DigestValue>
1159 </dsig:Reference>
1160 </dsig:SignedInfo>
1161 <dsig:SignatureValue>qnBvJoSgo4OoiYYaE3AwL5/EDq7BhTT6 ... QwllHK+zxy66I=</dsig:SignatureValue>
1162 <dsig:KeyInfo>
1163 <dsig:KeyName>C=CA, O=CPC, OU=EPM Service, CN=EPM Signature, E= ... </dsig:KeyName>
1164 <dsig:X509Data>
1165 <X509Certificate xmlns="http://www.w3.org/2000/09/xmldsig#">MIIEUDC ... EwZOBg==</X509Certificate>
1166 <X509SubjectName xmlns="http:// ... xmldsig#"> C=CA, O=CPC, OU=EPM Service, CN=EPM Signature, E=... </X509SubjectName>
1167 <X509IssuerSerial xmlns="http://www.w3.org/2000/09/xmldsig#">
1168 <X509IssuerName>C=CA, O=CPC, OU=EPM Service, CN=Electronic PostMark CA, E=... </X509IssuerName>
1169 <X509SerialNumber>25</X509SerialNumber>
1170 </X509IssuerSerial>
1171 </dsig:X509Data>
1172 </dsig:KeyInfo>
1173 <dsig:Object xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
1174 <dss:TstInfo xmlns:dss="http://www.docs.oasis-open.org/dss/2004/06/oasis-dss-1.0-core-schema-wd-25.xsd"
1175 <Id="TstInfo040327174718Z">
1176 <SerialNumber>1847365279</SerialNumber>
1177 <CreationTime>2004-03-27T17:47:18.750</CreationTime>
1178 <Policy/>
1179 <ErrorBound/>
1180 <Ordered/>
1181 <TSAX509SubjectName>C=CA, O=CPC, OU=EPM Service, CN=EPM Signature, ... </TSAX509SubjectName>
1182 </dss:TstInfo>
1183 </dsig:Object>
1184 <dsig:Object xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
1185 <epm:PostMarkedReceipt xmlns:epm="http://www.upu.int/EPMSservice" Id="Receipt040327174718Z">
1186 <Receipt>
1187 <TransactionKey>
1188 <Locator>CA</Locator>
1189 <Key>1234567890</Key>
1190 <Sequence>1</Sequence>
1191 </TransactionKey>
1192 <ServiceProviderURL>http://epmproxy.avalonworks.com/upubeta/</ServiceProviderURL>
1193 <TSAX509SubjectName>C=CA, O=CPC, OU=EPM Service, CN=EPM Signature, ... </TSAX509SubjectName>
1194 <TimeStampValue>040327174718Z</TimeStampValue>
1195 <RevocationStatusQualifier>CRL Checked</RevocationStatusQualifier>
1196 <ResourceID>12345678</ResourceID>
1197 </Receipt>
1198 <TimeStampToken MimeType="application/pkcs7-signature" />
1199 </epm:PostMarkedReceipt>

```

```

1200     </dsig:Object>
1201 </dsig:Signature>
1202 <!-- End of PostMarkedReceipt signature -->
1203 <!-- Beginning of signed document being PostMarked -->
1204     <Object Id="DetachedDataBeingSigned">
1205         <PersonalData>
1206             <Name>Ed Shallow</Name>
1207             <StreetAddress>1234 Mockingbird Lane</StreetAddress>
1208             <City>Yellowknife</City>
1209             <PostalCode>W1C6J3</PostalCode>
1210             <SocialInsuranceNumber>123456789</SIN>
1211         </PersonalData>
1212     </Object>
1213 <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" Id="TargetSignature">
1214     <dsig:SignedInfo>
1215         <dsig:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
1216         <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
1217         <dsig:Reference URI="#DetachedDataBeingSigned">
1218             <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1219             <dsig:DigestValue>Po3vwPXh8kdpRUazMGjzluao65I=</dsig:DigestValue>
1220         </dsig:Reference>
1221     </dsig:SignedInfo>
1222     <dsig:SignatureValue>KyKUMJKW .. Yi7swX0FjLkDDZNs=</dsig:SignatureValue>
1223 <dsig:KeyInfo>
1224     <dsig:KeyName>C=CA, O=Acme Corp, CN=Joe Public, E= ... </dsig:KeyName>
1225     <dsig:X509Data>
1226         <X509Certificate xmlns="http://www.w3.org/2000/09/xmldsig#">MIIE ... EwZOBg==</X509Certificate>
1227         <X509SubjectName> C=CA, O=Acme Corp, CN=Joe Public, E= ... </X509SubjectName>
1228         <X509IssuerSerial>
1229             <X509IssuerName>C=CA, O=Partner CA, O=For Test Use Only, CN=Partner CA, E= ... </X509IssuerName>
1230             <X509SerialNumber>25</X509SerialNumber>
1231         </X509IssuerSerial>
1232     </dsig:X509Data>
1233 </dsig:KeyInfo>
1234 </dsig:Signature>
1235 <!-- End of signed document being PostMarked -->
1236 </Document>

1237

1238

```


1239

7 Element cross-reference Table

1240 The following tables provide a summary of the Input elements, options, and corresponding Output elements for each of the usage scenarios. Comments
1241 are also provided.

1242 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 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		Affected	Elements	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 <code>Location</code> attribute is specified as standalone. If <code>Location</code> specifies embedded, the receipt will be embedded and returned in <code><DocumentWithSignature></code> . The <code>NodeName</code> element optionally controls the scope of the <code>PostMarkedReceipt</code> 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 <code><SignatureInfo></code> structure.	
	ReturnX509Info	O	✓	✓	Returns a <code><X509Info></code> structure.	
Output / Response Elements						
	Result	M	✓	✓		As per [DSSCore] . <code><ResultMajor></code> , <code><ResultMinor></code> , and <code><ResultMessage></code> will all be initialized and returned
	TransactionKey	M	✓	✓		Always initialized and returned.
	PostMarkedReceipt	O	✓	✓	See <code><IssuePostMarkedReceipt></code> above.	
	SignatureObject	O	✓		Initialized for CMS/PKCS7 signatures when <code><ReturnUpdatedSignature></code> has been specified. See also <code><DocumentWithSignature></code>	
	DocumentWithSignature	O		✓	Only initialized for XMLSig based signatures when <code><IssuePostMarkedReceipt></code> with a <code>Location</code> 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

1247

8 References

1248

8.1 Normative

1249

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

1250

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

1251

1252

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

1253

1254

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

1255

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

1256

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

1257

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

1258

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

1259

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

1260

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

1261

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

1262

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

1263

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

1264

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

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

1267 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be
1268 claimed to pertain to the implementation or use of the technology described in this document or the extent to
1269 which any license under such rights might or might not be available; neither does it represent that it has made any
1270 effort to identify any such rights. Information on OASIS's procedures with respect to rights in OASIS specifications
1271 can be found at the OASIS website. Copies of claims of rights made available for publication and any assurances
1272 of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the
1273 use of such proprietary rights by implementors or users of this specification, can be obtained from the OASIS
1274 Executive Director.

1275 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other
1276 proprietary rights which may cover technology that may be required to implement this specification. Please
1277 address the information to the OASIS Executive Director.

1278 Copyright © OASIS Open 2003. *All Rights Reserved.*

1279 This document and translations of it may be copied and furnished to others, and derivative works that comment
1280 on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in
1281 whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are
1282 included on all such copies and derivative works. However, this document itself does not be modified in any way,
1283 such as by removing the copyright notice or references to OASIS, except as needed for the purpose of
1284 developing OASIS specifications, in which case the procedures for copyrights defined in the OASIS Intellectual
1285 Property Rights document must be followed, or as required to translate it into languages other than English.

1286 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or
1287 assigns.

1288 This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL
1289 WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE
1290 USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES
1291 OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.