

## 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/>.

36 For information on whether any patents have been disclosed that may be essential to implementing this  
37 specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights  
38 section of the Technical Committee web page (<http://www.oasis-open.org/committees/dss/ipr.php>).  
39 The non-normative errata page for this specification is located at [http://www.oasis-](http://www.oasis-open.org/committees/dss/)  
40 [open.org/committees/dss/](http://www.oasis-open.org/committees/dss/).

---

41 **Notices**

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

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

53 Copyright © OASIS® 1993–2007.

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

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

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

67 The names "OASIS" are trademarks of OASIS, the owner and developer of this specification, and should be used  
68 only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and  
69 use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see  
70 <http://www.oasis-open.org/who/trademark.php> for above guidance.

---

# Table of Contents

71		
72	1	Introduction ..... 6
73	1.1	Terminology ..... 6
74	1.2	Normative References ..... 6
75	1.3	Non-Normative References ..... 6
76	1.4	Namespaces ..... 6
77	2	Profile Features ..... 8
78	2.1	Identifier ..... 8
79	2.2	Scope ..... 8
80	2.3	Relationship To Other Profiles ..... 8
81	2.4	Signature Objects ..... 8
82	2.5	Transport Binding ..... 8
83	2.6	Security Binding ..... 8
84	2.6.1	Security Requirements ..... 8
85	2.7	Common Elements ..... 9
86	2.7.1.1	Element <InputDocuments> ..... 9
87	2.7.1.2	Element <DocumentWithSignature> ..... 9
88	2.7.2	Element <PostMarkedReceipt> and the PostMarkedReceipt Signature ..... 9
89	2.7.3	Output Element <TransactionKey> ..... 11
90	2.7.4	Input Element <OrganizationID> ..... 12
91	3	Profile of Signing Protocol ..... 13
92	3.1	Element <SignRequest> ..... 13
93	3.1.1	Constraints on Element <OptionalInputs> ..... 13
94	3.1.1.1	Element SignatureType ..... 13
95	3.1.1.2	Element <KeySelector> ..... 13
96	3.1.1.3	Element <AddTimestamp> ..... 13
97	3.1.1.4	Optional Input <Properties> ..... 14
98	3.1.1.5	Optional Input <SignedReferences> ..... 14
99	3.1.1.6	Optional Input <IncludeObject> ..... 14
100	3.1.1.7	Optional Input <SignaturePlacement> ..... 14
101	3.1.2	EPM-specific <OptionalInputs> ..... 14
102	3.1.2.1	Element <DocumentContainsTemplate> ..... 14
103	3.1.2.2	Element <TransactionKey> ..... 15
104	3.1.2.3	Element <ClaimedIdentity> ..... 15
105	3.1.2.4	Element <OrganizationID> ..... 16
106	3.1.3	<OptionalInputs> Processing Directives ..... 16
107	3.1.3.1	Element <IssuePostMarkedReceipt> ..... 16
108	3.1.3.2	Element <StoreNonRepudiationEvidence> ..... 17
109	3.1.3.3	Element <ReturnSignatureInfo> ..... 17
110	3.1.3.4	Element <ReturnX509Info> ..... 17
111	3.2	Element <SignResponse> ..... 17
112	3.2.1	Element <Result> ..... 17
113	3.2.2	Element <SignatureObject> ..... 17
114	3.2.3	EPM-specific <OptionalOutputs> ..... 17
115	3.2.3.1	Element <TransactionKey> ..... 17
116	3.2.3.2	Element <PostMarkedReceipt> ..... 18
117	3.2.3.3	Element <SignatureInfo> ..... 18

118	3.2.3.4 Element <X509Info> .....	18
119	4 Profile of Verifying Protocol .....	20
120	4.1 Element <VerifyRequest>.....	20
121	4.1.1 Constraints on Element <OptionalInputs> .....	20
122	4.1.1.1 Element <InputDocuments>.....	20
123	4.1.1.2 SignatureObject .....	20
124	4.1.1.3 Element <AdditionalKeyInfo>.....	20
125	4.1.1.4 Element <ReturnProcessingDetails> .....	20
126	4.1.1.5 Element <ReturnSigningTime> .....	20
127	4.1.1.6 Element <ReturnSignerIdentity> .....	20
128	4.1.1.7 Element <VerifyManifests> .....	20
129	4.1.1.8 Element <ReturnUpdatedSignature> .....	20
130	4.1.1.9 Element <ReturnTransformedDocument>.....	20
131	4.1.2 EPM-specific <OptionalInputs>.....	21
132	4.1.2.1 Element <OrganizationID> .....	21
133	4.1.2.2 Element <IgnoreManifests> .....	21
134	4.1.2.3 Element <SignatureSelector> .....	21
135	4.1.2.4 Element <IssuePostMarkedReceipt> .....	22
136	4.1.3 <OptionalInputs> Processing Flags .....	23
137	4.1.3.1 Element <StoreNonRepudiationEvidence> .....	23
138	4.1.3.2 Element <ReturnSignatureInfo> .....	23
139	4.1.3.3 Element <ReturnX509Info>.....	23
140	4.2 Element <VerifyResponse>.....	23
141	4.2.1 Element <Result>.....	23
142	4.2.2 Element <SignatureObject> .....	23
143	4.2.3 Element <OptionalOutputs>.....	24
144	4.2.3.1 Element <DocumentWithSignature> .....	24
145	4.2.4 Element <EPM-specific OptionalOutputs> .....	24
146	4.2.4.1 Element <TransactionKey> .....	24
147	4.2.4.2 Element <PostMarkedReceipt>.....	24
148	4.2.4.3 Element <SignatureInfo>.....	24
149	4.2.4.4 Element <X509Info> .....	24
150	5 Signing Template Examples.....	25
151	6 PostMarkedReceipt Examples .....	29
152	7 Element cross-reference Table .....	35
153	A. Acknowledgements.....	41
154		

---

# 155 1 Introduction

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

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

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

## 166 1.1 Terminology

167 The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT",  
168 "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as described in IETF RFC  
169 2119 [RFC 2119]. These keywords are capitalized when used to unambiguously specify requirements over  
170 protocol features and behavior that affect the interoperability and security of implementations. When these words  
171 are not capitalized, they are meant in their natural-language sense.

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

## 174 1.2 Normative References

- 175 **[Core-XSD]** S. Drees et al. *DSS Schema*. OASIS, February 2007
- 176 **[DSSCore]** S. Drees et al. *Digital Signature Service Core Protocols and Elements*. OASIS, February 2007
- 177 **[RFC 2396]** S. Bradner. Key words for use in RFCs to Indicate Requirement Levels.  
178 <http://www.ietf.org/rfc/rfc2396.txt>, IETF RFC 2396, August 1998. .
- 179 **[TS 101733]** CMS Advanced Electronic Signatures. ETSI TS 101 733, January 2007.
- 180 **[XAdES]** XML Advanced Electronic Signatures. ETSI TS 101 903, March 2006
- 181 **[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.
- 182
- 183 **[XMLSig]** D. Eastlake et al. *XML-Signature Syntax and Processing*. <http://www.w3.org/TR/1999/REC-xml-names-19990114>, W3C Recommendation, February 2002.
- 184
- 185 **[RFC 2634]** P. Hoffman (ed.). Enhanced Security Services for S/MIME, <http://www.ietf.org/rfc/rfc2634.txt>,  
186 IETF RFC 2634 June 1999.
- 187 **[RFC 3369]** R. Housley, Message Syntax (CMS).. <http://www.ietf.org/rfc/rfc3369.txt> , IETF RFC 3369 August  
188 2002.
- 189 **[EPM]** Universal Postal Union, Electronic PostMark Web Service Description Language (WSDL) the  
190 UPU's Postal Technology Centre <http://www.ptc.upu.int/>

191

## 192 1.3 Non-Normative References

## 193 1.4 Namespaces

194 The structures described in this specification are contained in the schema file **[EPM]**. All schema listings in the

195 current document are excerpts from that schema file. In the case of a disagreement between the schema file and  
196 this document, the schema file takes precedence.

197 This schema is associated with the following XML namespace:

198 <http://www.docs.oasis-open.org/dss/2004/04/oasis-dss-1.0-profiles-EPM-wd-09#>

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

200 Conventional XML namespace prefixes are used in this document:

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

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

---

## 209 2 Profile Features

### 210 2.1 Identifier

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

### 212 2.2 Scope

213 This document profiles the DSS signing and verifying protocols defined in **[DSSCore]** and provides an OASIS  
214 DSS-compliant interface to selected services of the EPM. One of the primary intents of the EPM Profile is to  
215 simplify request and response processing for client callers by constraining **[DSSCore]** in several ways.

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

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

- 219 ➤ Easy to use EPM “Signing Templates”
- 220 ➤ PostMarked receipts
- 221 ➤ Same document signatures is the default and preferred mechanism for handling signature creation and  
222 verification
- 223 ➤ Certificate validation data
  - 224 ▪ Revocation references
  - 225 ▪ Certificate references
  - 226 ▪ Online Certificate Status Protocol (OCSP) responses
- 227 ➤ Timestamping from a CA-independent TimeStamp Authority

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

### 232 2.3 Relationship To Other Profiles

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

### 234 2.4 Signature Objects

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

### 237 2.5 Transport Binding

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

### 241 2.6 Security Binding

#### 242 2.6.1 Security Requirements

243 The TLS X.509 Server Authentication security binding as described in section 6.2.1 in **[DSSCore]** must be used.  
244 Although outside the scope of this protocol, clients are expected to authenticate to an implementation of this



245 specification. At a minimum HTTP Basic Authorization should be used to authenticate. Implementations are  
246 expected to validate the user and password contained in the HTTP header.

## 247 2.7 Common Elements

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

### 250 2.7.1.1 Element <InputDocuments>

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

256 When the <Document> element is passed in by the user of this profile, it is assumed that it contains only the  
257 content to be signed or the signed document to be verified. When users wish to use the EPM's "signing template"  
258 mechanism, they must also pass in a <DocumentContainsTemplate> element directive. Please also refer to  
259 section 3.1.2.1 below.

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

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

265 The <Document> element is also used to pass in a signature to be timestamped when the <SignatureType>  
266 specifies a timestamp type. The mimeType should specify application/pkcs7-signature when passing in  
267 a signature to be RFC 3161 timestamped.

### 268 2.7.1.2 Element <DocumentWithSignature>

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

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

```
278 <xs:element name="DocumentWithSignature">  
279   <xs:complexType>  
280     <xs:sequence>  
281       <xs:element ref="dss:Document"/>  
282     </xs:sequence>  
283   </xs:complexType>  
284 </xs:element>
```

### 285 2.7.2 Element <PostMarkedReceipt> and the PostMarkedReceipt Signature

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

292 specific context of EPM service provisioning. Semantics as follows:

293 ➤ **Sign**

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

296 ➤ **Verify**

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

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

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

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

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

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

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

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

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

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

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

```
323
324 <xs:element ref="epm:PostMarkedReceipt"
325
326 <!-- imported from the EPM schema -->
327 <xs:element name="PostMarkedReceipt" type="epm:PostMarkedReceiptType">
328
329 <xs:complexType name="PostMarkedReceiptType">
330   <xs:sequence>
331     <xs:choice>
332       <xs:element name="PKCS7SignedReceipt" type="epm:PKCS7SignedReceiptType"/>
333       <xs:element name="XMLSignedReceipt" type="epm:QualifiedDataType"/>
334     </xs:choice>
335   </xs:sequence>
336 </xs:complexType>
337
338 <xs:complexType name="PKCS7SignedReceiptType">
339   <xs:sequence>
340     <xs:element name="Receipt" type="epm:ReceiptType"/>
341     <xs:element name="ReceiptSignature" type="epm:QualifiedDataType"
342 nillable="true"/>
343   </xs:sequence>
```

```

344 </xs:complexType>
345 <xs:complexType name="ReceiptType">
346   <xs:sequence>
347     <xs:element name="TransactionKey" type="epm:TransactionKeyType"/>
348     <xs:element name="Requester" type="xs:string"/>
349     <xs:element name="Operation" type="xs:string"/>
350     <xs:element name="TSAX509SubjectName" type="xs:string"/>
351     <xs:element name="TimeStampValue" type="xs:string"/>
352     <xs:element name="RevocationStatusQualifier" type="xs:string"/>
353     <xs:element name="TimeStampToken" type="epm:QualifiedDataType" nillable="true"
354 minOccurs="0" maxOccurs="1"/>
355     <xs:element name="MessageImprint" type="xs:base64Binary" nillable="true"/>
356     <xs:element name="PostMarkImage" type="epm:QualifiedDataType" nillable="true"/>
357     <xs:element name="ReceiptMetadata" type="epm:ReceiptMetadataType"
358 nillable="true" minOccurs="0" maxOccurs="unbounded"/>
359   </xs:sequence>
360 </xs:complexType>
361
362 <xs:complexType name="ReceiptMetadataType">
363   <xs:sequence>
364     <xs:element name="Name" type="xs:string"/>
365     <xs:choice>
366       <xs:element name="Value" type="xs:string"/>
367       <xs:element name="EncodedValue" type="epm:QualifiedDataType"/>
368     </xs:choice>
369   </xs:sequence>
370 </xs:complexType>
371

```

372

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

376 **Note 2:** The binary <TimeStampToken> element above can be omitted for [XMLSig]-based  
377 <SignatureType>'s since the PostMarkedReceipt is itself a signature which covers the <TstInfo>  
378 structure. EPM implementations using TimeStamp Authorities (TSAs), are however free to initialize this element  
379 with an RFC3161 Timestamp Token if they wish. The example in section 6 does not initialize the  
380 <TimeStampToken> element.

### 381 2.7.3 Output Element <TransactionKey>

382 This complexType is a compound key made up of 3 elements uniquely identifying each event in the an EPM  
383 Lifecycle. The EPM generates and returns a new and unique <TransactionKey> with all response operations.  
384 The <Locator> element is used to identify the particular EPM instance when multiple EPM instances are  
385 involved, as is the case with cross-border transactions. Please refer to EPM documentation for usage guidelines.

```

386 <xs:element ref="epm:TransactionKey"
387
388 <!-- imported from the EPM schema -->
389 <xs:element name="TransactionKey" type="epm:TransactionKeyType">
390 <xs:complexType name="TransactionKeyType">
391   <xs:sequence>
392     <xs:element name="Locator" type="epm:LocatorType"/>
393     <xs:element name="Key" type="xs:string"/>
394     <xs:element name="Sequence" type="xs:positiveInteger"/>
395   </xs:sequence>
396 </xs:complexType>
397 <xs:complexType name="LocatorType">
398   <xs:sequence>
399     <xs:element name="CountryCode" type="xs:string"/>
400     <xs:element name="Version" type="xs:string"/>
401     <xs:element name="ServiceProvider" type="xs:string" nillable="true"/>

```

```
402     <xs:element name="Environment" type="xs:string" nillable="true"/>
403     </xs:sequence>
404 </xs:complexType>
```

405

## 406 2.7.4 Input Element <OrganizationID>

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

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

412

---

## 413 3 Profile of Signing Protocol

### 414 3.1 Element <SignRequest>

#### 415 3.1.1 Constraints on Element <OptionalInputs>

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

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

##### 420 3.1.1.1 Element SignatureType

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

422 The following <SignatureType> URNs are supported:

423     ➤ Signature creation URNs:

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

426     ➤ Timestamp creation URNs:

- 427         ▪ oasis:names:tc:dss:1.0:core:schema:XMLTimeStampToken
- 428         ▪ urn:ietf:rfc:3161 (i.e. a CMS/PKCS7 timestamp token)

429 The first 2 URNs instruct the EPM to create a signature. The last 2 URNs instruct the EPM to create a timestamp.  
430 The context and processing rules within which the EPM creates signatures is different than the context within  
431 which the EPM creates timestamps. These differences will be highlighted below as they apply to each optional  
432 input and output, as constrained by the <SignatureType> chosen above. If no restriction is mentioned below,  
433 one may assume that the optional input is valid for timestamp <SignatureType>'s as well.

##### 434 3.1.1.2 Element <KeySelector>

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

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

##### 445 3.1.1.3 Element <AddTimestamp>

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

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

452 The `<AddTimestamp>` is also supported on the Verify operation, and will update the signature being verified.  
453 See also `<IssuePostMarkedReceipt>` which returns a timestamped receipt structure and has different  
454 semantic meaning. Please refer to section 4 covering the Verify.

455 **Note:** Content timestamps, created before signature generation, are currently not supported in the EPM profile  
456 (e.g. a timestamp created before signature generation and referenced as a signed property or attribute). They  
457 may however be added in a subsequent release of the EPM profile.

#### 458 3.1.1.4 Optional Input `<Properties>`

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

#### 461 3.1.1.5 Optional Input `<SignedReferences>`

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

#### 464 3.1.1.6 Optional Input `<IncludeObject>`

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

#### 470 3.1.1.7 Optional Input `<SignaturePlacement>`

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

### 475 3.1.2 EPM-specific `<OptionalInputs>`

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

#### 478 3.1.2.1 Element `<DocumentContainsTemplate>`

479 The `<DocumentContainsTemplate>` optional input element is a directive which tells the implementation that  
480 the `<Document>` element passed in on the request is a "signing template". It is used when users elect to utilize  
481 the EPM's "signing template" mechanism. EPM-supported signing templates contain not only the data to be  
482 signed, but also the format and directives of the signature to be created, expressed as valid [XMLSig] elements.  
483 In this fashion more elaborate signatures involving transforms, signed and unsigned properties, manifests, and  
484 multiple `<Reference>` elements can be supported without complex XML request constructs. [XMLSig] elements  
485 such as `<SignatureValue>`, `<DigestValue>`, and `<X509Certificate>` are populated by the EPM  
486 service based on the template provided. The user leaves these crypto-specific element tags empty, and the EPM  
487 service will automatically include the generated content and return the signed document in the  
488 `<DocumentWithSignature>` element of the `<SignResponse>`. See Example 1 in section 5 for an example of  
489 signing templates. More details are available in the EPM Systems Integrator's Guide and other EPM  
490 documentation available through the UPU.

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

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

494

### 495 3.1.2.2 Element <TransactionKey>

496 Please refer to the description in section 2.7.3 entitled [Element <TransactionKey>](#)

### 497 3.1.2.3 Element <ClaimedIdentity>

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

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

```
515
516 <xs:element name="ClaimedIdentity">
517   <xs:complexType>
518     <xs:sequence>
519       <xs:element name="Name" type="saml:NameIdentifierType"/>
520       <xs:element ref="epm:SupportingInfo"/>
521     </xs:sequence>
522   </xs:complexType>
523 </xs:element>
524
525 <!-- imported from the EPM schema -->
526 <xs:element name="SupportingInfo" type="epm:SupportingInfoType"/>
527 <xs:complexType name="SupportingInfoType">
528   <xs:element name="BasicAuth" type="epm:BasicAuthType" nillable="true"/>
529   <xs:element name="RequesterSignature" type="epm:QualifiedDataType" nillable="true"/>
530   <xs:element name="AlternateIdentity" type="epm:AlternateIdentityType" nillable="true"/>
531 </xs:complexType>
532
533 <xs:complexType name="epm:QualifiedDataType">
534   <xs:simpleContent>
535     <xs:extension base="xs:base64Binary">
536       <xs:attribute name="MimeType" type="xs:string"/>
537     </xs:extension>
538   </xs:simpleContent>
539 </xs:complexType>
540
541 <xs:complexType name="BasicAuthType">
542   <xs:sequence>
543     <xs:element name="UserID" type="xs:string"/>
544     <xs:element name="Password" type="xs:string" nillable="true"/>
545   </xs:sequence>
546 </xs:complexType>
547
548 <xs:complexType name="AlternateIdentityType" abstract="true">
549   <xs:sequence>
550     <xs:element name="IdentityToken" type="xs:anyType"/>
551   </xs:sequence>
552 </xs:complexType>
```

553

### 554 3.1.2.4 Element <OrganizationID>

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

### 556 3.1.3 <OptionalInputs> Processing Directives

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

#### 560 3.1.3.1 Element <IssuePostMarkedReceipt>

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

- 567 ➤ For a Location attribute value of standalone regardless of the <SignatureType>, processing is as  
568 follows:
- 569 ▪ The <PostMarkedReceipt> XML element will be returned as a standalone optional output  
570 structure as defined in section 2.7.2. Standalone <PostMarkedReceipt>'s are self-contained  
571 and contain a timestamp signature which binds the receipt to the signature value of the signature  
572 being created as part of this Sign operation.
- 573 ➤ For a Location attribute value of embedded and a <SignatureType> value of urn:ietf:rfc:3275 (i.e.  
574 XMLSig), processing is as follows:
- 575 ▪ The EPM will first create an [XMLSig] based “detached” signature covering the input document.  
576 The input document's contents will be outside the produced signature and referenced by it. The  
577 EPM will then add a <PostMarkedReceipt> detached signature structure covering the  
578 <SignatureValue> of the first signature just created. The resulting signed and PostMarked  
579 document will be returned in the <DocumentWithSignature> element.
- 580 ➤ A Location attribute value of embedded with a <SignatureType> value of urn:ietf:rfc:3369 (i.e.  
581 CMS/PKCS7) is not supported.
- 582 ▪ A signature timestamp (i.e. an RFC 3161 timestamptoken) however can be embedded in a  
583 CMS/PKCS7 signature by using the <AddTimestamp> optional input described in section  
584 3.1.1.3. This timestamp bears the Issuer name of the Post's TimeStamp Authority.

585

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

```
587 <xs:element ref="epm:IssuePostMarkedReceipt">
588
589 <!-- imported from the EPM schema -->
590 <xs:element name="IssuePostMarkedReceipt" type="epm:IssuePostMarkedReceiptType">
591 <xs:complexType name="IssuePostMarkedReceiptType">
592   <xs:sequence>
593     <xs:element name="Location" type="epm:ValidLocation" minOccurs="0"/>
594     <xs:element name="PostMarkImage" type="epm:PostMarkImageType" minOccurs="0"/>
595   </xs:sequence>
596 </xs:complexType>
597
598 <xs:complexType name="PostMarkImageType">
599   <xs:simpleContent>
600     <xs:extension base="xs:boolean">
601       <xs:attribute name="Format" type="xs:string" default="JPG"/>

```



```
602     <xs:attribute name="Size" type="epm:ValidImageSize" default="Small" />
603     </xs:extension>
604   </xs:simpleContent>
605 </xs:complexType>
606
```

### 607 **3.1.3.2 Element <StoreNonRepudiationEvidence>**

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

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

611

### 612 **3.1.3.3 Element <ReturnSignatureInfo>**

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

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

617

### 618 **3.1.3.4 Element <ReturnX509Info>**

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

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

623

## 624 **3.2 Element <SignResponse>**

### 625 **3.2.1 Element <Result>**

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

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

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

### 629 **3.2.2 Element <SignatureObject>**

630 If successful, the server will return a <ds:Signature> with the signature properties as defined in [DSSCore].  
631 Location of the generated signature will be determined based on signature type and envelope type. All  
632 CMS/PKCS7 signatures will be returned in the <SignatureObject> element. [XMLSig] based enveloping  
633 signatures will also be returned in the <SignatureObject> element. Enveloped and detached signatures are  
634 returned in the <DocumentWithSignature> element described below.

### 635 **3.2.3 EPM-specific <OptionalOutputs>**

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

#### 638 **3.2.3.1 Element <TransactionKey>**

639 Please refer to section 3.1.2.2 for a description of how the <TransactionKey> element which is used on both  
640 input and on output as an identification and retrieval mechanism and to support subsequent reference to specific

641 service calls.

### 642 3.2.3.2 Element <PostMarkedReceipt>

643 If the <IssuePostMarkedReceipt> optional input is included, then this optional output will be returned. It is  
644 essentially a standalone receipt represented as an enveloping signature. See section 2.7.2 for details.

### 645 3.2.3.3 Element <SignatureInfo>

646 This structure can be returned on both the Sign and Verify operations and is returned whenever the  
647 <ReturnSignatureInfo> element is included in the request. Together with the <X509Info> element these  
648 elements provide more detail on the signature just created or being verified. The element <SignedContent> is  
649 used in conjunction with the Verify operation and allows users to extract the signed content from a CMS/PKCS7  
650 signature thus alleviating the need to parse the ASN.1 structure. See <VerifyResponse> below. The  
651 <SignedContent> element on a CMS/PKCS7 Sign response is empty as the user has just passed this content  
652 in to be signed, however the <SignedContent> element on an XMLDSig Sign request will contain the  
653 transformed content as it existed prior to digest calculation for the user's reference.

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

```
656 <xs:element ref="epm:SignatureInfo"  
657  
658 <!-- imported from the EPM schema -->  
659 <xs:element name="SignatureInfo" type="epm:SignatureInfoType">  
660 <xs:complexType name="SignatureInfoType">  
661 <xs:sequence>  
662 <xs:element name="SignedContent" type="epm:QualifiedDataType" nillable="true"/>  
663 <xs:element name="ContentHash" type="xs:string" nillable="true"/>  
664 <xs:element name="ContentHashAlgo" type="xs:string" nillable="true"/>  
665 <xs:element name="ContentEncryptAlgo" type="xs:string" nillable="true"/>  
666 <xs:element name="SigningTime" type="xs:string" nillable="true"/>  
667 <xs:element name="PKCS1" type="epm:QualifiedDataType" nillable="true"/>  
668 </xs:sequence>  
669 </xs:complexType>
```

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

### 671 3.2.3.4 Element <X509Info>

672 This structure is returned whenever the <ReturnX509Info> element is included in the request. The  
673 <X509ValidationData> element is the default implementation of the abstract base type called  
674 <GenericValidationDataType> and contains a signed OCSF Validation Data element returned by the EPM.  
675 This element attests to the validity of the certificate used in the signing operation. It will contain signed content as  
676 per RFC 2560 and also described in RFC 3126 as returned by a standard OCSF Responder. If an EPM DSS  
677 implementation is not using an OCSF responder, then sufficient certificate chain and revocation references must  
678 be included here possibly as an alternate implementation of the abstract base type. Additionally, many  
679 jurisdictions (e.g. the EU) require that this validation info be signed by the Certified Service Provider (CSP). This  
680 is not an issue when using an RFC 2560-compliant OCSF Responder. Definitions of the other elements are  
681 standard certificate fields and descriptions are also available in the EPM Systems Integrator's Guide.

```
682 <xs:element ref="epm:X509Info"  
683  
684 <!-- imported from the EPM schema -->  
685 <xs:element name="X509Info" type="epm:X509InfoType">  
686 <xs:complexType name="X509InfoType">  
687 <xs:sequence>  
688 <xs:element name="X509Subject" type="xs:string"/>  
689 <xs:element name="X509Issuer" type="xs:string" nillable="true"/>  
690 <xs:element name="X509Serial" type="xs:string" nillable="true"/>  
691 <xs:element name="X509StatusSource" type="xs:string"/>  
692 <xs:element name="X509ValidFrom" type="xs:string"/>
```

```

693     <xs:element name="X509ValidTo" type="xs:string"/>
694     <xs:element name="X509Certificate" type="xs:string" nillable="true"/>
695     <xs:element name="X509RevocationReason" type="xs:string" nillable="true"/>
696     <xs:element name="X509RevocationReasonString" type="xs:string" nillable="true"/>
697     <xs:element name="X509RevocationTime" type="xs:string" nillable="true"/>
698     <xs:element name="X509ValidationData" type="epm:X509ValidationDataType"
699 nillable="true"/>
700     </xs:sequence>
701 </xs:complexType>
702
703 <xs:complexType name="GenericValidationDataType" abstract="true">
704     <xs:sequence>
705         <xs:element name="GenericValidationData" type="xs:anyType"/>
706     </xs:sequence>
707 </xs:complexType>
708
709 <xs:complexType name="X509ValidationDataType">
710     <xs:complexContent>
711         <xs:extension base="epm:GenericValidationDataType">
712             <xs:sequence>
713                 <xs:element name="X509ValidationData"
714 type="epm:QualifiedDataType"/>
715             </xs:sequence>
716         </xs:extension>
717     </xs:complexContent>
718 </xs:complexType>
719

```

720

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

722

---

## 723 4 Profile of Verifying Protocol

### 724 4.1 Element <VerifyRequest>

#### 725 4.1.1 Constraints on Element <OptionalInputs>

##### 726 4.1.1.1 Element <InputDocuments>

727 Must be initialized when users wish to verify [XMLSig] based enveloped or detached signatures which contain the  
728 signed content. Presently constrained to one <Document> occurrence. This single <Document> must contain  
729 signature(s) to be verified as well as all signed content referenced by the signature(s) as part of a “same  
730 document” signature.

##### 731 4.1.1.2 SignatureObject

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

##### 734 4.1.1.3 Element <AdditionalKeyInfo>

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

##### 736 4.1.1.4 Element <ReturnProcessingDetails>

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

##### 738 4.1.1.5 Element <ReturnSigningTime>

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

##### 742 4.1.1.6 Element <ReturnSignerIdentity>

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

##### 745 4.1.1.7 Element <VerifyManifests>

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

##### 749 4.1.1.8 Element <ReturnUpdatedSignature>

750 This optional input is not supported by the EPM. The only signature update supported is when the caller specifies  
751 <AddTimestamp>. This produces an embedded timestamp similar to the one produced as part of the Sign  
752 protocol and described in section 3.1.1.3 entitled Element <AddTimestamp>. The RFC3161 compliant timestamp  
753 token is included in the signature as an unauthenticated attribute of the verified signature. This conventionally  
754 timestamped CMS/PKCS7 signature, now updated, will be returned in the <SignatureObject>.

##### 755 4.1.1.9 Element <ReturnTransformedDocument>

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

## 757 4.1.2 EPM-specific <OptionalInputs>

### 758 4.1.2.1 Element <OrganizationID>

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

### 760 4.1.2.2 Element <IgnoreManifests>

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

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

### 763 4.1.2.3 Element <SignatureSelector>

764 This optional `SignatureSelector` element qualifies the XMLDSIG signature(s) to be verified by the EPM. This  
765 element may also serve useful if the user is unsure of exactly what has been verified, and wishes to control the  
766 verification process more explicitly.

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

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

773 .

```
774 <xs:element name="SignatureSelector" type="epm:SignatureSelectorType" />
775   <xs:complexType name="SignatureSelectorType">
776     <xs:sequence>
777       <xs:choice>
778         <xs:element name="NodeName" type="xs:string" minOccurs="1"
779 maxOccurs="unbounded" />
780         <xs:element name="XPathSelector" type="epm:XPathSelectorType" />
781       </xs:choice>
782     </xs:sequence>
783   </xs:complexType>
784
785   <xs:complexType name="XPathSelectorType">
786     <xs:sequence>
787       <xs:element name="XPath" type="xs:string" />
788       <xs:element name="Namespace" type="xs:string" nillable="true" />
789       <xs:element name="Qualifier" type="xs:string" nillable="true" />
790     </xs:sequence>
791   </xs:complexType>
```

792  
793 *EXAMPLE* The user would specify string values of `lg1:Party1` and/or `lg1:Party2` to explicitly instruct the  
794 EPM what to Verify. By default the EPM will search for signature nodes specified as `<dsig:Signature>`, which  
795 appear as descendants of the document root.

```
796 <lg1:Party1>
797   <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmlsig#">
798     ...
799   </dsig:Signature>
800 </lg1:Party1>
801 <lg1:Party2>
802   <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmlsig#">
803     ...
804   </dsig:Signature>
805 </lg1:Party2>
```

806

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

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

```
812 <lgl:Document xmlns:lgl="http://www.lgl.org/SomeService"  
813           xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">  
814   <lgl:Signatures>  
815     <dsig:Signature>1st</dsig:Signature>  
816     ...  
817     <dsig:Signature>2nd</dsig:Signature>  
818     ...  
819     <dsig:Signature>3rd</dsig:Signature>  
820     ...  
821   </lgl:Signatures>  
822 </lgl:Document>
```

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

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

827 In both examples a value of http://www.lgl.org/SomeService and lgl should be specified for the  
828 NameSpace and Qualifier elements respectively in order to allow the XPath string expression to evaluate.  
829

#### 830 **4.1.2.4 Element <IssuePostMarkedReceipt>**

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

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

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

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

839     ▪ The <PostMarkedReceipt> XML element will be returned as a standalone optional output  
840     structure as defined in section 2.7.2. Standalone <PostMarkedReceipt>'s are self-contained  
841     and contain a timestamp signature which binds the receipt to the signature value of the signature  
842     being verified as part of this Verify operation.

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

846 The incoming signed document will contain an **[XMLSig]** based "detached" signature covering the  
847 required content within the input document. The input document's signed content will be outside the  
848 signature and referenced by it. The EPM will verify this signature. If the signature(s) verify successfully,  
849 the EPM will then add a <PostMarkedReceipt> detached signature structure covering the  
850 <SignatureValue>'s of the signature(s) just verified.

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

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

855     ▪ A signature timestamp (i.e. an RFC 3161 timestamp token) however can be embedded in a

856 CMS/PKCS7 signature by using the <AddTimestamp> optional input described in section  
857 3.1.1.3. This timestamp bears the Issuer name of the Post's TimeStamp Authority.

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

```
859 <xs:element ref="epm:IssuePostMarkedReceipt">
860
861 <!-- imported from the EPM schema -->
862 <xs:complexType name="IssuePostMarkedReceiptType">
863   <xs:sequence>
864     <xs:element name="Location" type="epm:ValidLocation" minOccurs="0"/>
865     <xs:element name="PostMarkImage" type="epm:PostMarkImageType" minOccurs="0"/>
866   </xs:sequence>
867 </xs:complexType>
868
869 <xs:complexType name="PostMarkImageType">
870   <xs:simpleContent>
871     <xs:extension base="xs:boolean">
872       <xs:attribute name="Format" type="xs:string" default="JPG"/>
873       <xs:attribute name="Size" type="epm:ValidImageSize" default="Small"/>
874     </xs:extension>
875   </xs:simpleContent>
876 </xs:complexType>
877
```

878

### 879 4.1.3 <OptionalInputs> Processing Flags

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

#### 883 4.1.3.1 Element <StoreNonRepudiationEvidence>

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

#### 885 4.1.3.2 Element <ReturnSignatureInfo>

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

#### 887 4.1.3.3 Element <ReturnX509Info>

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

### 889 4.2 Element <VerifyResponse>

#### 890 4.2.1 Element <Result>

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

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

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

#### 894 4.2.2 Element <SignatureObject>

895 This element is only returned when the <AddTimestamp> optional input is included. Please refer to section  
896 4.1.1.8 for details.

## 897 **4.2.3 Element <OptionalOutputs>**

### 898 **4.2.3.1 Element <DocumentWithSignature>**

899 If the <IssuePostMarkedReceipt> optional input is included and its Location attribute specifies embedded,  
900 then this optional output will be returned. See the scenario described in the 2<sup>nd</sup> bullet within section 4.1.2.4 above  
901 for more details.

## 902 **4.2.4 Element <EPM-specific OptionalOutputs>**

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

### 905 **4.2.4.1 Element <TransactionKey>**

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

### 908 **4.2.4.2 Element <PostMarkedReceipt>**

909 If the <IssuePostMarkedReceipt> optional input is included in the Verify request and its Location attribute  
910 specifies standalone, then this optional output will be returned. It is essentially a standalone receipt signature.  
911 See also section 0 above.

### 912 **4.2.4.3 Element <SignatureInfo>**

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

### 914 **4.2.4.4 Element <X509Info>**

915 See section 3.2.3.4 for a detailed explanation of this elements usage.



916

## 5 Signing Template Examples

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

919

### 920 **Example 1:**

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

925

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

```
927 <DocumentWithTemplate/>
```

```
928 <Document>
```

```
929   <Data>
```

```
930     <SubData1>
```

```
931       <SubSubData1 MimeType="text/plain">This is the data to be signed.</SubSubData1>
```

```
932       <SubSubData2 MimeType="text/plain">This is the data to be signed.</SubSubData2>
```

```
933       <SubSubData3 MimeType="text/plain">This is the data to be signed.</SubSubData3>
```

```
934     </SubData1>
```

```
935     <SubData2>This is the data to be signed.</SubData2>
```

```
936     <SubData3>This is the data to be signed.</SubData3>
```

```
937   </Data>
```

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

```
939     <SignedInfo>
```

```
940       <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
```

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

```
942         <Reference URI="">
```

```
943           <Transforms>
```

```
944             <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
```

```
945           </Transforms>
```

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

```
947           <DigestValue></DigestValue>
```

```
948         </Reference>
```

```
949       </SignedInfo>
```

```
950       <SignatureValue>
```

```
951     </SignatureValue>
```

```
952     <KeyInfo>
```

```
953       <KeyName>C=CA, S=Ontario, L=Ottawa, O=CPC, OU=eServices, CN=Ed Test, E=ed.shallow@rogers.com</KeyName>
```

```
954       <X509Data>
```

```
955         <X509Certificate></X509Certificate>
```

```
956         <X509SubjectName></X509SubjectName>
```

```
957         <X509IssuerSerial>
```

```
958         </X509IssuerSerial>
959     </X509Data>
960 </KeyInfo>
961 </Signature>
962 </Document>
```

963

## 964 **Example 2:**

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

```
967
968 <?xml version="1.0" encoding="UTF-8"?>
969 <DocumentWithTemplate/>
970 <Document>
971     <Data>
972         <SubData1>
973             <SubSubData1 MimeType="text/plain">This is the data to be signed.</SubSubData1>
974             <SubSubData2 MimeType="text/plain">This is the data to be signed.</SubSubData2>
975             <SubSubData3 MimeType="text/plain">This is the data to be signed.</SubSubData3>
976         </SubData1>
977         <SubData2>This is data.</SubData2>
978         <SubData3>This is data.</SubData3>
979     </Data>
980     <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
981         <SignedInfo>
982             <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
983             <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
984                 <Reference URI="#xpointer(/Document/Data/SubData1)">
985                     <Transforms>
986                         <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
987                     </Transforms>
988                     <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
989                     <DigestValue></DigestValue>
990                 </Reference>
991             </SignedInfo>
992             <SignatureValue>
993             </SignatureValue>
994             <KeyInfo>
995                 <X509Data>
996                     <X509Certificate></X509Certificate>
997                     <X509SubjectName></X509SubjectName>
998                     <X509IssuerSerial>
999                     </X509IssuerSerial>
1000                 </X509Data>
1001             </KeyInfo>
1002         </Signature>
1003 </Document>
```

1004  
1005  
1006  
1007  
1008  
1009  
  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049

### Example 3:

This is a more complicated example using intersect and subtract XPath Filters. This 3<sup>rd</sup> 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:KeyInfo>
  </dsig:Signature>
</Document>
```

```
1050         <dsig:X509SubjectName></dsig:X509SubjectName>
1051         <dsig:X509IssuerSerial></dsig:X509IssuerSerial>
1052     </dsig:X509Data>
1053 </dsig:KeyInfo>
1054 </dsig:Signature>
1055 </Document>
1056
```

1057

## 6 PostMarkedReceipt Examples

1058 PostMarked receipts are normally returned to the application as standalone XML structures, whether they are of type CMS/PKCS7 or XMLSig.  
1059 Upon request however <PostMarkedReceipt>'s can be embedded in the incoming signed document. This is true for both the Sign protocol as  
1060 well as the Verify protocol. The first example below is a standalone <PostMarkedReceipt>, and the second example is one that is embedded  
1061 into the signed document.

### 1062 Example 1 Standalone PostMarkedReceipt:

1063 This is an example of a PostMarkedReceipt. It is essentially a conventional XMLSig enveloping signature over the <SignatureValue> of the  
1064 target signature being PostMarked. It contains three (3) <Reference> elements pointing to each of the following:

- 1065 ➤ a standard <dss:TstInfo> as per [DSSCore]
- 1066 ➤ an <epm:PostMarkedReceipt> element from the [EPM] schema
- 1067 ➤ the <SignatureValue> element of the target signature being PostMarked

1068 Selected element contents have been deliberately truncated for brevity and clarity.

```
1069 <?xml version="1.0" encoding="UTF-8"?>
1070 <dsig:Signature Id="PostMarkedReceiptSignature" xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
1071   <dsig:SignedInfo>
1072     <dsig:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1073     <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
1074     <dsig:Reference URI="#TstInfo">
1075       <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1076       <dsig:DigestValue>jWkUFR6epvkrTaxTiQ33DiWy+l8=</dsig:DigestValue>
1077     </dsig:Reference>
1078     <dsig:Reference URI="#Receipt">
1079       <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1080       <dsig:DigestValue>9JWkDlh/8Cs9Slu2QmZixOJl+x0=</dsig:DigestValue>
1081     </dsig:Reference>
1082     <dsig:Reference URI="#PostMarkedSignatures">
1083       <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1084       <dsig:DigestValue>MOBYPfrllMBcJz6yobjhrwH9KP4=</dsig:DigestValue>
1085     </dsig:Reference>
1086   </dsig:SignedInfo>
1087   <dsig:SignatureValue>qnBvJoSgo4OoiYyaE3AwL5/EDq7BhTT6 ... QwllHK+zxy66I=</dsig:SignatureValue>
1088   <dsig:KeyInfo>
1089     <dsig:KeyName>C=CA, O=CPC, OU=EPM Service, CN=EPM Signature, E= ... </dsig:KeyName>
1090     <dsig:X509Data>
1091       <X509Certificate xmlns="http://www.w3.org/2000/09/xmldsig#">MIIEUDC ... EwZOBg==</X509Certificate>
1092     <X509SubjectName xmlns="http:// ... xmldsig#"> C=CA, O=CPC, OU=EPM Service, CN=EPM Signature, E=... </X509SubjectName>
1093     <X509IssuerSerial xmlns="http://www.w3.org/2000/09/xmldsig#">
1094       <X509IssuerName>C=CA, O=CPC, OU=EPM Service, CN=Electronic PostMark CA, E=... </X509IssuerName>
```

```

1096         <X509SerialNumber>25</X509SerialNumber>
1097     </X509IssuerSerial>
1098 </dsig:X509Data>
1099 </dsig:KeyInfo>
1100 <dsig:Object xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
1101     <dss:TstInfo xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema" Id="TstInfo">
1102         <SerialNumber>1847365279</SerialNumber>
1103         <CreationTime>2004-03-27T17:47:18.750</CreationTime>
1104         <Policy/>
1105         <ErrorBound/>
1106         <Ordered/>
1107         <TSA>C=CA, O=CPC, OU=EPM Service, CN=EPM Signature, E= ... </TSA>
1108     </dss:TstInfo>
1109 </dsig:Object>
1110 <dsig:Object xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" Id="Receipt">
1111     <epm:PostMarkedReceipt xmlns:epm="http://www.upu.int/EPMSservice/schemas">
1112         <Receipt>
1113             <TransactionKey>
1114                 <Locator>
1115                     <CountryCode>CA</CountryCode>
1116                     <Version>114</Version>
1117                     <ServiceProvider>ePost Corporation</ServiceProvider>
1118                     <Environment xsi:nil="true"/>
1119                 </Locator>
1120                 <Key>1234567890</Key>
1121                 <Sequence>1</Sequence>
1122             </TransactionKey>
1123             <Requester>CN=Joe Public, O=VeriSign Class 1 Certificate, C=CA, E=joe.public@rogers.com</Requester>
1124             <Operation>Verify</Operation>
1125             <TSAX509SubjectName> ... </TSAX509SubjectName>
1126             <MessageImprint> ... </MessageImprint>
1127             <PostMarkImage> ... </PostMarkImage>
1128             <RevocationStatusQualifier>CRL Checked</RevocationStatusQualifier>
1129             <TimeStampToken mimeType="application/pkcs7-signature"></TimeStampToken>
1130             <ReceiptMetadata>
1131                 <Name> ... </Name>
1132                 <Value>... </Value>
1133             </ReceiptMetadata>
1134         </Receipt>
1135     </epm:PostMarkedReceipt>
1136 </dsig:Object>
1137 <dsig:Object xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
1138     <epm:PostMarkedContent xmlns:epm="http://www.upu.int/EPMSservice/schemas" Id="PostMarkedSignatures">
1139         <epm:PostMarkedSignatureValue>1NiHC2bBKfT ... AlfhecQo</epm:PostMarkedSignatureValue>
1140     </epm:PostMarkedContent>
1141 </dsig:Object>
1142 </dsig:Signature>
1143
1144 If the standalone PostMarkedReceipt covers more than one signature, the 3rd Referenced Object would look like this:

```

```

1145
1146 <dsig:Object xmlns:dsig=http://www.w3.org/2000/09/xmldsig# Id="PostMarkedSignatures">
1147   <epm:PostMarkedContent xmlns:epm="http://www.upu.int/EPMSchema/schemas">
1148     <PostMarkedSignatureValue>1NiHC2bBKfT ... AlfcGhecQo=</PostMarkedSignatureValue>
1149     <PostMarkedSignatureValue>aqw95gB/Tz5 ... n0qRqMHJ5c=</PostMarkedSignatureValue> ...
1150     ... would include as many other PostMarkedSignatureValue elements as may be present in the PostMarked document
1151     ...
1152   </epm:PostMarkedContent>
1153 </dsig:Object>
1154
1155

```

1156 **Note:** Similarly, when the <PostMarkedReceipt>'s signature scope simply covers data (as opposed to a SignatureValue), then the 3<sup>rd</sup>  
1157 <Reference> will be to an <Object> containing the hash of the data to be PostMarked with base64 encoding specified.

```

1158   <dsig:Object xmlns:dsig=http://www.w3.org/2000/09/xmldsig# Id="PostMarkedData">
1159     <epm:PostMarkedContent xmlns:epm="http://www.upu.int/EPMSchema/schemas"
1160       Encoding="http://www.w3.org/2000/09/xmldsig#base64">RGF0YSBgdG8gcQ...gVGkgMTUgMTI6MTA=</PostMarkedContent>
1161   </dsig:Object>

```

1162

### 1163 **Example 2 Embedded PostMarkedReceipt:**

1164 This is an example of an embedded <PostMarkedReceipt> returned after a successful Verify operation. It is a conventional XMLSig detached  
1165 signature over the <SignatureValue> of the target signature(s) being PostMarked. It contains three (3) <Reference> elements pointing to  
1166 each of the following:

- 1167 ➤ a standard <dss:TstInfo> as per **[DSSCore]**
- 1168 ➤ an <epm:PostMarkedReceipt> element from the **[EPM]** schema
- 1169 ➤ the <SignatureValue> element of the target signature(s) being PostMarked

1170 Note that depending on the value of the optional SignatureSelector element within the Verify request, the <PostMarkedReceipt> can  
1171 potentially cover all <SignatureValue>'s in the signed document when the document contains multiple signatures.

1172 Selected element contents have been deliberately truncated for brevity and clarity.

```

1173
1174 <?xml version="1.0" encoding="UTF-8"?>
1175 <!DOCTYPE Document [
1176   <!ATTLIST Object Id ID #IMPLIED>
1177 ]>
1178 <Document>
1179 <!-- Beginning of PostMarkedReceipt signature -->
1180   <dsig:Signature Id="PostMarkedReceiptSignature" xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
1181     <dsig:SignedInfo>
1182       <dsig:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1183       <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
1184       <dsig:Reference URI="#TstInfo">

```

```

1185         <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1186         <dsig:DigestValue>3Lk/6TE7ldqeXZFUJ9qqaPInm24=</dsig:DigestValue>
1187     </dsig:Reference>
1188     <dsig:Reference URI="#Receipt">
1189         <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1190         <dsig:DigestValue>430zTvcoa9r8Rpr5DiVZf7IPvl8=</dsig:DigestValue>
1191     </dsig:Reference>
1192     <dsig:Reference URI="">
1193         <dsig:Transforms>
1194             <dsig:Transform Algorithm="http://www.w3.org/TR/1999/REC-xpath-19991116">
1195                 <dsig:XPath xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
1196                     ancestor-or-self::dsig:SignatureValue[../@Id!="PostMarkedReceiptSignature"]
1197                 </dsig:XPath>
1198             </dsig:Transform>
1199         </dsig:Transforms>
1200         <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1201         <dsig:DigestValue>LRAX6mCfAq8hprb8UMU1H35PTYw=</dsig:DigestValue>
1202     </dsig:Reference>
1203 </dsig:SignedInfo>
1204 <dsig:SignatureValue>qnbVJoSgo4OoiYyaE3AwBL5/EDq7BhTT6 ... Qw11HK+zxy66I=</dsig:SignatureValue>
1205 <dsig:KeyInfo>
1206     <dsig:KeyName>C=CA, O=CPC, OU=EPM Service, CN=EPM Signature, E= ... </dsig:KeyName>
1207     <dsig:X509Data>
1208         <X509Certificate xmlns="http://www.w3.org/2000/09/xmldsig#">MIIEUDC ... EwZOBg==</X509Certificate>
1209 <X509SubjectName xmlns="http:// ... xmldsig#"> C=CA, O=CPC, OU=EPM Service, CN=EPM Signature, E=... </X509SubjectName>
1210 <X509IssuerSerial xmlns="http://www.w3.org/2000/09/xmldsig#">
1211     <X509IssuerName>C=CA, O=CPC, OU=EPM Service, CN=Electronic PostMark CA, E=... </X509IssuerName>
1212     <X509SerialNumber>25</X509SerialNumber>
1213 </X509IssuerSerial>
1214 </dsig:X509Data>
1215 </dsig:KeyInfo>
1216 <dsig:Object xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" Id="TstInfo">
1217     <dss:TstInfo xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema">
1218         <SerialNumber>1847365279</SerialNumber>
1219         <CreationTime>2004-03-27T17:47:18.750</CreationTime>
1220         <Policy/>
1221         <ErrorBound/>
1222         <Ordered/>
1223         <TSAX509SubjectName>C=CA, O=CPC, OU=EPM Service, CN=EPM Signature, ... </TSAX509SubjectName>
1224     </dss:TstInfo>
1225 </dsig:Object>
1226 <dsig:Object xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" Id="Receipt">
1227     <epm:PostMarkedReceipt xmlns:epm="http://www.upu.int/EPMServices/schemas">
1228         <Receipt>
1229             <TransactionKey>
1230                 <Locator>
1231                     <CountryCode>CA</CountryCode>
1232                     <Version>114</Version>
1233                     <ServiceProvider>ePost Corporation</ServiceProvider>

```



```

1234         <Environment xsi:nil="true" />
1235     </Locator>
1236     <Key>1234567890</Key>
1237     <Sequence>1</Sequence>
1238 </TransactionKey>
1239 <Requester>CN=Joe Public, O=VeriSign Class 1 Certificate, C=CA, E=joe.public@rogers.com</Requester>
1240 <Operation>Verify</Operation>
1241 <TSAX509SubjectName> ... </TSAX509SubjectName>
1242 <MessageImprint> ... </MessageImprint>
1243 <PostMarkImage> ... </PostMarkImage>
1244 <RevocationStatusQualifier>CRL Checked</RevocationStatusQualifier>
1245 <TimeStampToken MimeType="application/pkcs7-signature"></TimeStampToken>
1246 <ReceiptMetadata>
1247     <Name> ... </Name>
1248     <Value> ... </Value>
1249 </ReceiptMetadata>
1250 </Receipt>
1251 </epm:PostMarkedReceipt>
1252 </dsig:Object>
1253 </dsig:Signature>
1254 <!-- End of PostMarkedReceipt signature -->
1255 <!-- Beginning of signed document being PostMarked -->
1256 <Object Id="DetachedDataBeingSigned">
1257     <PersonalData>
1258         <Name>Ed Smith</Name>
1259         <StreetAddress>1234 Mockingbird Lane</StreetAddress>
1260         <City>Yellowknife</City>
1261         <PostalCode>W1C6J3</PostalCode>
1262         <SocialInsuranceNumber>123456789</SIN>
1263     </PersonalData>
1264 </Object>
1265 <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" Id="TargetSignature">
1266     <dsig:SignedInfo>
1267         <dsig:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1268         <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
1269         <dsig:Reference URI="#DetachedDataBeingSigned">
1270             <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1271             <dsig:DigestValue>Po3vwPXh8kdpRUAzMGjzlua065I=</dsig:DigestValue>
1272         </dsig:Reference>
1273     </dsig:SignedInfo>
1274     <dsig:SignatureValue>KyKUMJKW ... Yi7swX0FjLkDDZNs=</dsig:SignatureValue>
1275     <dsig:KeyInfo>
1276         <dsig:KeyName>C=CA, O=Acme Corp, CN=Joe Public, E= ... </dsig:KeyName>
1277         <dsig:X509Data>
1278             <X509Certificate xmlns="http://www.w3.org/2000/09/xmldsig#">MIIE ... EwZOBg==</X509Certificate>
1279             <X509SubjectName> C=CA, O=Acme Corp, CN=Joe Public, E= ... </X509SubjectName>
1280             <X509IssuerSerial>
1281                 <X509IssuerName>C=CA, O=Partner CA, O=For Test Use Only, CN=Partner CA, E= ... </X509IssuerName>
1282                 <X509SerialNumber>25</X509SerialNumber>

```

```
1283         </X509IssuerSerial>
1284         </dsig:X509Data>
1285     </dsig:KeyInfo>
1286 </dsig:Signature>
1287 <!-- End of signed document being PostMarked -->
1288 </Document>
1289
```

1290

## 7 Element cross-reference Table

1291

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.

1292

1293

### Sign Protocol

	Optionality	As Used in Sig Type		Elements Affected	Comments
		CMS/PKCS7	XMLSIG		
<b>Input / Request Elements</b>					
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 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 <code>&lt;PostMarkedReceipt&gt;</code> 's can be controlled via the <code>Locator</code> attribute.
	DocumentContainsTemplate	O		✓	Signatures produced will be returned in <code>&lt;DocumentWithSignature&gt;</code>	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.
<b>Processing Option Flags</b>						
	AddTimestamp	O	✓			Attribute not req'd. Produces a conventional timestamp as opposed to a <code>&lt;PostMarkedReceipt&gt;</code> .
	IssuePostMarkedReceipt	O	✓		Returns a standalone <code>&lt;PostMarkedReceipt&gt;</code> element.	
	IssuePostMarkedReceipt	O		✓	Returns a standalone <code>&lt;PostMarkedReceipt&gt;</code> 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>&lt;DocumentWithSignature&gt;</code> .	
	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.	
<b>Output / Response Elements</b>						
	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.	

1294  
1295  
1296  
1297  
1298

**Verify Protocol**

	Optionality	As Used in Sig Type		Elements Affected	Comments
		CMS/PKCS7	XMLSIG		
<b>Input / Request Elements</b>					
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.
<b>Processing Option Flags</b>					
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.	
<b>Output / Response Elements</b>						
	Result	M	✓	✓		As per <b>[DSSCore]</b> , <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.	

1299



---

1300 **A. Acknowledgements**

1301 The following individuals have participated in the creation of this specification and are gratefully acknowledged:

1302 **Participants:**

1303 Trevor Perrin, individual

1304 Nick Pope, Thales eSecurity

1305 Juan Carlos Cruellas, Centre d'aplicacions avançades d'Internet (UPC)

1306