Digital Signatures for XML

Richard D. Brown – GlobeSet, Inc.

Revision History

16-June-98:

Initial Draft

13-October-98:

Revision to reflect 1998-09-16 XML Namespaces proposal

04-April-99:

Fix numerous typos,

Complete the algorithm chapters,

Add a Comments and Suggestions chapter,

Insert the DTD definition,

Add a Resources element – CS #99122601

Document Status

This document, file name draft-brown-xml-sig-01.doc is intended to become a Proposed Standard RFC. Distribution of this document is unlimited. Comments should be sent to the XML DSIG mailing list <xml-dsig@socratic.org> or to the author.

Europe), munnari.oz.au (Pacific Rim), ftp.ietf.org (US East Coast), or ftp.isi.edu (US West Coast).

Contents

Introduction

XML, the Extensible Markup Language [x], is a syntactical standard elaborated by the World Wide Web Consortium. Subset of an international text-processing standard known as SGML (Standard Generalized Markup Language), XML is intended primarily for structuring data exchanged and served over the World Wide Web.

Structuring information permits data to be easily read, exchanged, and acted upon by software agents. It is a first step toward the production of a Web of machine-readable semantics. But, the usefulness of such structured information is limited if its authenticity and trustworthiness cannot be verified. The Web cannot suffice with XML - it needs Signed-XML.

Objective and Requirements

The objective of this specification is to define syntax and procedures for the computation, verification, and encoding of digital signatures using XML. It proposes a solution to authenticating Web resources by means of XML.

This specification has been established in light of the requirements that have been gathered while reviewing diverse projects and alternative proposals such as IOTP [x], BIPS [x], SDML [x], FSML [x] and XMLDSIG [x]. Previous experiences with binary cryptographic syntaxes such as PKCS#7 [x] and CMS [x] have also played an important role in this specification.

The redaction of this specification has been driven by the following requirements:

- The solution shall provide a means for building authentication into XML applications, but shall also propose an XML alternative to binary signature syntax for signing arbitrary contents.

- The solution shall provide for digital signature and message authentication codes, considering symmetric and asymmetric authentication schemes as well as dynamic establishment of keying material.
- The solution shall provide for certificate-based and account-based authentication schemes.
- The solution shall provide a mechanism that eases the production of composite documents that consist of the combination by addition or deletion of authenticated blocks of information, while preserving verifiability of the origin and authenticity of these blocks of information.
- The solution shall enable authentication of part or totality of a Web document.
- The solution shall enable authentication of internal and external resources.
- The solution shall provide for extended signature functionality such as co-signature, endorsement, plurality of recipients, etc.

Signature Basics

Signature Element

This specification consists primarily of the definition of an XML element known as the Signature element. This element is comprised of two sub-elements. The first one is a set of authenticated attributes, known as the signature **Manifest**, which comprises such things as a unique reference to the resources being authenticated and an indication of the keying material and algorithms being used. The second sub-element consists of the digital signature value.

```
<Signature>
    <Manifest>
        (resources information block)
        (other attributes)
        (originator information block)
        (recipient information block)
        (key agreement algorithm information block)
        (signature algorithm information block)
    </Manifest>
    <Value encoding='encoding scheme'>
        (encoded signature value)
    </Value>
</Signature>
```

The digital signature is not computed directly from the pieces of information to be authenticated. Instead, the digital signature is computed from a set of authenticated attributes (the Manifest), which includes a reference to, and a digest of, these pieces of information. The authentication is therefore *'indirect'*.

Resources Element

The Resources element consists of a collection of Resource elements that, in turn, consist of a unique and unambiguous reference to a resource being authenticated. Each Resource element is constructed of a locator, a fingerprint, and optionally a content-type qualifier.

```
<Resources>
    <Resource>
        <Locator href='resource locator'/>
        <ContentInfo type='type qualifier'/>
        <Digest>
            (digest information block)
        </Digest>
    </Resource>
    <Resource>
        ...
    </Resource>
</Resources>
```

The resource locator is implemented as a simple XML Link [x]. This not only provides a unique addressing scheme for internal and external resources, but also facilitates authentication of composite documents.

Other Attributes Element

The Attributes element consists of a collection of Attribute elements that enable attachment and authentication of specific pieces of information that relate to a given signature. An Attribute element is constructed of a type, a criticality, and a value.

```
<Attributes>
    <Attribute type='signing-time' critical='true'>
        <Date value='1998-10-28T08:15-0500' />
    </Attribute>
    <Attribute type='private-type' critical='false'>
        (ANY attribute value)
    </Attribute>
</Attributes>
```

The attribute value consists of ANY content that is defined in the application DTD. Nevertheless, to facilitate the adoption of ’standard’ attributes, the Signature DTD provides for common types such as ’*signing-time*.’

Originator and Recipient Information Elements

The purpose of the Originator and Recipient information elements consists of providing identification and keying material for these respective parties.

```
<OriginatorInfo>
    (identification information block)
    (keying material information block)
</OriginatorInfo>

<RecipientInfo>
    (identification information block)
    (keying material information block)
</RecipientInfo>
```

The actual content of these two elements depends on the authentication scheme being used and the existence or non-existence of a prior relationship between the parties. In some circumstances, it may be quite difficult to distinguish between identification and keying material information. A unique reference to a digital certificate provides for both. This may also stand true for an account number when a prior relationship exists between the parties.

The Originator information element is mandatory. Depending on the existence or non-existence of a prior relationship with the recipient, this block either refers to a public credential such as a digital certificate or displays a unique identifier known by the recipient.

The Recipient information element may be used when a document contains multiple signature information blocks, each being intended for a particular recipient. A unique reference in the Recipient information block helps the recipients identify their respective Signature information block.

The Recipient information element may also be used when determination of the authentication key consists of a combination of keying material provided by both parties. This would be the case, for example, when establishing a key by means of Diffie Hellman [x] Key Exchange algorithm.

Key Agreement Algorithm Element

The Key Agreement Algorithm element is an optional element that could be used to indicate the algorithm to be used for deriving a one-time session key from a master key. Usage of one-time session key prevents some kinds of attack that require a large volume of cipher-text to be produced by a given key.

```
<KeyAgreementAlgorithm>
    (algorithm information block)
</KeyAgreementAlgorithm>
```

Signature Algorithm Element

The Signature Algorithm element indicates the algorithm to be used for computation of the signature value.

```
<SignatureAlgorithm>
    (algorithm information block)
</SignatureAlgorithm>
```

In consideration of the requirements stated previously, this document uses the terminology of ’signature’ for qualifying indifferently signature and authentication schemes. Therefore, the signature algorithm mentioned above might refer to a signature algorithm such as DSS or to a message authentication code (MAC) such as HMAC.

Signature Principles

Enabling Signature in XML Applications

As mentioned previously, this specification provides among others a means for building authentication into XML applications. The mechanism adopted herein considers the *’XML Namespaces’ s*pecification [x], which defines the requirements for combining multiple DTDs or parts of individual DTD into a single document.

According to this specification, an XML application can build digital signature support by referring explicitly to the elements defined in the Signature DTD. This is accomplished by associating a namespace prefix to the Signature DTD and qualifying Signature element names by means of this prefix.

Association of a namespace prefix to a DTD shall be done by means of a *xmlns* attribute, which could appear in any element that either refers to or contains sub-elements that refer to elements of the DTD considered. A qualified name consists of a namespace prefix, a colon, and a name.

```
<MyDocument xmlns:dsig='Signature-DTD-URI'>

```

```
    <MyElement id='authenticated-infos'>
        ...
    </MyElement>

    <dsig:Signature>
        <dsig:Manifest>
            <dsig:Resources>
                <dsig:Resource>
                    <dsig:Locator href='authenticated-infos'/>
                </dsig:Resource>
            </dsig:Resources>
            ...
        </dsig:Manifest>
        <dsig:Value>
            ...
        </dsig:Value>
    </dsig:Signature>

</MyDocument>
```

Encapsulating Arbitrary Contents

To facilitate encapsulation of arbitrary contents into an XML document, the Signature DTD defines a Package element. Quite similar to a MIME wrapper, this element provides for such things as content type and content encoding.

```
<dsig:Package>
    <dsig:ContentInfo type='type qualifier'/>
    <dsig:Value encoding='encoding scheme'>
        (safe content)
    </dsig:Value>
</dsig:Package>
```

Implementing Endorsement

Endorsement consists of signing another signature. To facilitate endorsement, the definition of the Signature element provides for an element identifier attribute, which can be used to target a Signature element from a Resource element.

```
<dsig:Signature id='signature'>
    <dsig:Manifest>
        <dsig:Resources>
            <dsig:Resource>
                <dsig:Locator href='resource locator'/>
                ...
            </dsig:Resource>
        </dsig:Resources>
        ...
    </dsig:Manifest>
    ...
</dsig:Signature>

<dsig:Signature id='counter-signature'>
    <dsig:Manifest>
        <dsig:Resources>
            <dsig:Resource>
                <dsig:Locator href='signature'/>
            ...
            </dsig:Resource>
        </dsig:Resources>
        ...
    </dsig:Manifest>
    ...
</dsig:Signature>
```

Supporting Composite Documents

Some protocols consist of the exchange of documents that result from the combination by addition or deletion of common information blocks. The current proposal shall preserve verifiability of the origin and authenticity of these blocks of information as they are exchanged between parties.

To facilitate authentication of such composite documents, this specification has adopted an 'indirect' authentication scheme – the signature is applied to unambiguous references to the resources being authenticated instead of their contents. Signature verification does not require the actual contents of the resources.

This indirect scheme can be further extended when multiple signatures must be produced for a large number of resources – repeating the resource elements in multiple signature Manifests might not be optimal. In such circumstances, the application DTD can leverage the Resources element to share the resource definitions between multiple signature elements.

```
<dsig:Resources id='shared=resources'>
    <dsig:Resource>
        ...
    </dsig:Resource>
    <dsig:Resource>
        ...
    </dsig:Resource>
    ...
</dsig:Resources>

<dsig:Signature>
    <dsig:Manifest>
        <dsig:Resources>
```

```
            <dsig:Resource>
                <dsig:Locator href='shared-resources'>
                ...
            </dsig:Resource>
        </dsig:Resources>
        ...
    </dsig:Manifest>
    ...
</dsig:Signature>

<dsig:Signature>
    <dsig:Manifest>
        <dsig:Resources>
            <dsig:Resource>
                <dsig:Locator href='shared-resources'>
                ...
            </dsig:Resource>
        </dsig:Resources>
        ...
    </dsig:Manifest>
    ...
</dsig:Signature>
```

The adoption of simple XML links as resource locators makes possible the authentication of composite documents. If IDREFs were used instead, it would have been impossible to ensure validity of partial documents - some IDREFs could have been left referencing non-embedded IDs.

Facilitating One-pass Processing

Without further definitions, it would be impossible to determine which blocks of information require authentication and which algorithms need to be employed before interpretation of the Resource elements. These elements being generally located at the end of the document, this restriction would prevent computation of the digests during acquisition of the blocks of information.

To facilitate one-pass processing, the current specification uses another functionality offered by the namespaces proposal. This functionality provides for the definition of *global attributes* that may be used and recognized across multiple elements. This document specifies the **dsig:eval** global attribute, which could be used for identifying the blocks of information to be authenticated.

The dsig:eval attribute shall refer to an Algorithm element or list of Algorithm elements that identify the algorithms and parameters to be used for computation of the digest of the element being authenticated. To comply with the requirements of one-pass processing, the Algorithm element should be declared before making use of the dsig:eval attribute.

```
<dsig:DigestAlgorithms id='digest-algorithm'>
    <Algorithm id='SHA1' type='urn:nist-gov:sha1'\>
    <Algorithm id='MD5' type='urn:rsasdi-com:MD5'\>
</dsig:DigestAlgorithms>

<MyElement id='authenticated-infos'
    dsig:eval='SHA1 MD5'>
    ...
</MyElement>

<dsig:Signature>
    <dsig:Manifest>
        <dsig:Resources>
            <dsig:Resource>
                <dsig:Locator href='authenticated-infos'>
                <Digest>
                    <Algorithm id='SHA1' type='urn:nist-
gov:sha1'\>
                    <Value encoding='base64'>
                        ANBbdshh456wh5==
                    </Value>
            </dsig:Resource>
        </dsig:Resources>
        ...
    </dsig:Manifest>
    ...
</dsig:Signature>
```

When encountering the dsig:eval global attribute on an element, the XML parser is immediately aware of the requirement of computing the digest or digests of this element. All the pieces of information necessary for such computation are provided by the Algorithm element or elements referenced by the attribute.

The dsig:eval attribute is purely declarative. Discrepancies between the dsig:eval attribute and the digest algorithm definition in the Resource element shall not invalidate the signature. At the most, such discrepancies will result in a verification failure if the signature-agent cannot memorize nor rewind its input stream.

Syntax Comments

Namespace Attributes

All the elements defined by the Signature DTD are explicitly bound to the XMLDSIG namespace by means of a *dsig* prefix. In order to make sure that every element could be individually imported by other XML applications, the element definitions given hereinafter systematically declare a fixed *xmlns:dsig* attribute.

```
<!ELEMENT dsig:element definition...>

<!ATTLIST dsig:element
      xmlns:dsig  CDATA       #FIXED      %xmldsig.dtd;
      ...
>
```

Recall that many XML applications, presumably including namespaces-sensitive ones, fail to require validating processors. For correct operations with such applications, namespaces declarations must be also provided either directly or via default attributes declared in the internal subset of the DTD.

dsig:eval Global Attribute

As mentioned previously, this specification defines a dsig:eval global attribute that could be used for identifying a block of information to be authenticated. This attribute shall refer to an Algorithm element or elements, which should be declared before making use of the attribute.

The XML Namespaces specifications do not explicitly provide for declaration of global attributes. Distinguishing between global attributes and element attributes exists only in the prose description of such attributes. An essential property of global attributes consists nonetheless of the uniqueness of their name that is independent of the elements where they are defined.

The definition of elements that could be subject to authentication may define the dsig:eval attribute as follows:

```
<!ELEMENT element definition...>

<!ATTLIST element
      dsig:eval   IDREFS      #IMPLIED
>
```

Recall that the namespace prefix that is bound to the XMLDSIG namespace shall be defined before being employed. However, such definition may occur in the element that defines the dsig:eval attribute.

The reader shall notice that the terminology 'dsig:eval' is inappropriate and used solely for illustrative purposes. This simply means that the name of this attribute is eval and it belongs to the XMLDSIG namespace (whatever prefix is used).

Uniform Resource Names

To prevent potential name conflicts in the definition of the numerous type qualifiers considered herein, this specification uses Uniform Resource Names

[RFC 2141]. Nonetheless, this specification leverages established standards such as MIME types by providing unambiguous mapping conventions.

A complete list of proposed URNs is given in appendix. This list is temporary and will be submitted for approval to the authors or promoters of the algorithms and data types referenced by these URNs.

Basic Data Types

To facilitate the adoption of common procedures for the encoding of attribute and parameter values, this specification defines a series of elements not directly mandated by the Signature DTD. These definitions propose a common approach to encoding basic data types such as Integer, Float, Date, etc… It is expected that these definitions will be reconsidered, as the results of other W3C Activities in this area (i.e. XML-Data) will be adopted.

Algorithm Definitions

This specification adopts a unique Algorithm data type. Though noticeably different from its ASN1 counterpart, this data type serves a similar purpose and provides for the definition of algorithm-specific parameters.

The most noticeable difference with ASN1 consists of the assimilation of sub-algorithms as parameters of the primary algorithm. In other words, where ASN1 recognizes an algorithm of the type AlgxWithAlgy (i.e. DsaWithSha1) the current specifications recognize Algx with an Algy parameter.

This recursive construct is expected to be more versatile and shall provide a solution applicable to the definition of algorithms in general. However, this definition does not preclude the adoption of shortcuts such as the ones proposed by ASN1. It does not preclude either the adoption of default parameter values.

Detailed Signature Syntax

Algorithm

The Algorithm element consists of a basic data type that uniquely identifies a given algorithm and indicates the parameter values to be used during computation. The construct is recursive and allows a parameter value to refer to another Algorithm element.

```
<!ELEMENT dsig:Algorithm ( dsig:Parameter* )>
```

```
<!ATTLIST dsig:Algorithm
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      id                ID          #IMPLIED
      type              CDATA       #REQUIRED
>
```

Content Description

Parameter:	The contents of an Algorithm element consists of an optional collection of Parameter elements, which are specified on a per algorithm basis.

Attributes Description

id:	Element identifier that could be used for referencing this element (from a dsig:eval global attribute for example).
type:	The type of the algorithm expressed as a Uniform Resource Name.

Attribute

The Attribute element consists of a complementary piece of information, which shall be included in the authenticated part of the document. Though this specification defines standard attributes, this element has been defined primarily for enabling some level of customization in the signature element. An Attribute element consists of a value, a type, and a criticality.

```
<!ELEMENT dsig:Attribute ANY>

<!ATTLIST dsig:Attribute
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      type              NMTOKEN     #REQUIRED
      critical    ( true | false )  #IMPLIED    ’false’
>
```

Content Description

ANY:	The actual value of an attribute depends solely upon its type.

Attributes Description

type:	Type of the attribute expressed as a Uniform Resource Name.
critical:	Boolean value that indicates if the attribute is critical (true) or not (false). A recipient shall reject a signature that contains a critical attribute that he does not recognize. However, an unrecognized non-critical attribute may be ignored.

Attributes

The Attributes element consists of a collection of complementary attributes, which shall be included in the authenticated part of the document.

```
<!ELEMENT dsig:Attributes ( dsig:Attribute+ )>

<!ATTLIST dsig:Attributes
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
>
```

Content Description

Attribute:	Collection of Attribute elements.

Certificate

The Certificate element may be used for either providing the value of a digital certificate or specifying a location from where it may be retrieved.

```
<!ELEMENT dsig:Certificate (
      dsig:IssuerAndSerialNumber,
      ( dsig:Value | dsig:Locator )
)>

<!ATTLIST dsig:Certificate
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      type              NMTOKEN     #REQUIRED
>
```

Content Description

IssuerAndSerialNumber:	Unique identifier of this certificate. This element has been made mandatory is order to prevent unnecessary decoding during validation of a certificate chain. This feature also helps certificates caching, especially when the value is not directly provided.
Value:	Encoding of the certificate value. The actual value to be encoded depends upon the type of the certificate.
Locator:	XML link element that could be used for retrieving a copy of the digital certificate. The actual value being returned by means of this locator depends upon the protocol being used.

Attributes Description

type:	Type of the digital certificate expressed as a Uniform Resource Name.

Certificates

The Certificates element consists of a collection of Certificate elements. The Certificate elements contained in this element are intended to be sufficient to make chains from the originator credential(s) to a recognized ’certification authority’ for all the recipients. However, this element may contain more Certificate elements than necessary or, alternatively, less than necessary if it is known that recipients have an alternate means of obtaining necessary certificates.

```
<!ELEMENT dsig:Certificates ( dsig:Certificate+ )>

<!ATTLIST dsig:Certificates
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
>
```

Content Description

Certificate:	A collection of Certificate elements.

ContentInfo

The purpose of the ContentInfo element is to describe a given content such that a receiving user agent can deal with the data in an appropriate manner.

```
<!ELEMENT dsig:ContentInfo EMPTY>

<!ATTLIST dsig:ContentInfo
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      type              CDATA       #REQUIRED
      subtype           CDATA       #IMPLIED
>
```

Attributes Description

type:	Type of the content expressed as a Uniform Resource Name.
subtype:	Optional sub-classing of the content type.

Date

The Date element consists of a constrained ISO 8601:1998 date and time value.

```
<!ELEMENT dsig:Date EMPTY>

<!ATTLIST dsig:Date
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      value             CDATA       #REQUIRED
>
```

Attributes Description

value:	The date value expressed according to the format defined below.

Date Format

This specification requires date values to be expressed according to the following pattern:

```
YYYY '-' MM '-' DD 'T' hh ':' mm [':' ss ['.' f+]]('+' | '-') zzzz
```

YYYY:	four-digit year
MM:	two-digit month (01=January, etc.)
DD:	two-digit day of the month (01-31)
hh:	two digits of hour (00-23)
mm:	two digits of minute (00-59)
ss:	two digits of second (00-59) optional
f:	digit(s) of fractions of second - optional
zzzz:	four digits of amount of offset from UTC expressed in hour (00-11) and minute (00-59)

For example, ’1994-11-05T16:15:02.031-0500’ denotes November 5, 1994, 4:15:02 pm and 31 milliseconds, US Eastern Standard Time.

Digest

The Digest element consists of the fingerprint of a given resource. This element is constructed of two sub-elements. This first one indicates the algorithm to be used for computation of the fingerprint. The second element consists of the fingerprint value.

```
<!ELEMENT dsig:Digest ( dsig:Algorithm, dsig:Value )>

<!ATTLIST dsig:Digest
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
>
```

Content Description

Algorithm:	Algorithm to be used for computation of the digest value.
Value:	Digest value after proper encoding.

DigestAlgorithms

The DigestAlgorithms element consists of a collection of Algorithm elements that define the algorithms and parameter values to be employed in the computation of digest values. It is primarily used along with the dsig:eval global attribute for enabling one-pass processing.

```
<!ELEMENT dsig:DigestAlgorithms ( dsig:Algorithm+ )>

<!ATTLIST dsig:DigestAlgorithms
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
>
```

Content Description

Algorithm:	A collection of digest algorithm definitions.

Identifier

The Identifier element enables identification between parties that benefit from a prior relationship. The actual meaning and content of this element is left to the parties.

```
<!ELEMENT dsig:Identifier EMPTY>

<!ATTLIST dsig:Identifier
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      value             CDATA       #REQUIRED
>
```

Attributes Description

value:	Identification data value.

Integer

The Integer element is a primary data type that is used in the definition of algorithm parameters.

```
<!ELEMENT dsig:Integer EMPTY>

<!ATTLIST dsig:Integer
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      value             CDATA       #REQUIRED
>
```

Attributes Description

value:	Value of the element given according to the format given below.

Integer Format

This specification requires integer values to be expressed according to the following pattern:

```
['+'|'-'] n+
```

For example, +128, -35635, and 64535 are valid integer values.

IssuerAndSerialNumber

The IssuerAndSerialNumber element identifies a certificate, and thereby an entity and a public key, by the distinguished name of the certificate issuer and an issuer-specific certificate serial number.

```
<!ELEMENT dsig:IssuerAndSerialNumber EMPTY>

<!ATTLIST dsig:IssuerAndSerialNumber
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      issuer            CDATA       #REQUIRED
      number            CDATA       #REQUIRED
>
```

Attributes Description

issuer:	Distinguished name of the issuing certification authority.
number:	Issuer-specific certificate serial number.

KeyAgreementAlgorithm

The KeyAgreementAlgorithm element specifies the algorithm to be employed for establishment of a one-time session key.

```
<!ELEMENT dsig:KeyAgreementAlgorithm ( dsig:Algorithm )>

<!ATTLIST dsig: KeyAgreementAlgorithm
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
```

```
>
```

Content Description

Algorithm:	Algorithm and parameters to be used for establishment of the session key.

Keyword

The Keyword element is a primary data type that is used in the definition of algorithm parameters.

```
<!ELEMENT dsig:Keyword EMPTY>

<!ATTLIST dsig:Keyword
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      value             CDATA       #REQUIRED
>
```

Attributes Description

value:	Value of the element given as a free form string.

Locator

The Locator element consists of simple XML link [XLink]. This element allows unambiguous reference to a resource or fragment of a resource.

```
<!ELEMENT dsig:Locator EMPTY>

<!ATTLIST dsig:Locator
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      xml:link          CDATA       #FIXED      'simple'
      href              CDATA       #REQUIRED
>
```

Attributes Description

xml:link:	Required XML link attribute that specifies the nature of the link (simple in this case).

href:	Locator value that may contains either a URI [RFC 2396], a fragment identifier, or both.

Manifest

The Manifest element consists of a collection of attributes that specify such things as a unique reference to the resource being authenticated and an indication of the keying material and algorithms to be used. The signature value is actually computed from the Manifest.

```
<!ELEMENT dsig:Manifest (
      dsig:Resources, dsig:Attributes?,
      dsig:OriginatorInfo, dsig:RecipientInfo?,
      dsig:KeyAgreementAlgorithm?, dsig:SignatureAlgorithm
)>

<!ATTLIST dsig:Manifest
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
>
```

Content Description

Resources:	A collection of Resource elements that consist of a unique and unambiguous reference to the resources being authenticated.
Attributes:	Optional element that consists of a collection of complementary attributes to be authenticated.
OriginatorInfo:	Element that provides identification and keying material information related to the originator.
RecipientInfo:	Optional element that provides identification and keying material information related to the recipient.
KeyAgreementAlgorithm:	Optional element that indicates the algorithm to be used for establishment of a one-time session key.
SignatureAlgorithm:	Algorithm to be used for computation of the signature value.

OriginatorInfo

The OriginatorInfo element is used for providing identification and keying material information for the originator.

```
<!ELEMENT dsig:OriginatorInfo ANY>

<!ATTLIST dsig:OriginatorInfo
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
>
```

<u>Content Description</u>

ANY:	Identification and keying material information may consist of ANY construct. Such a definition allows the adoption of application-specific schemes. However, implementations that comply with the current DTD MUST be able to recognize and process the elements Identifier and IssuerAndSerialNumber defined below.

Package

The Package element enables encapsulation of an arbitrary content into an XML document. Behaving like a MIME wrapper, the Package element provides for such things as content type identification and content encoding.

```
<!ELEMENT dsig:Package (
      dsig:ContentInfo?, dsig:Value
)>

<!ATTLIST dsig:Package
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      dsig:eval         IDREF       #IMPLIED
      id                ID          #IMPLIED
>
```

<u>Content Description</u>

ContentInfo:	Element that provides type information regarding the content of the Package.
Value:	Element that displays the content value of the Package and provides information regarding possible encoding.

Attributes Description

id:	Element identifier that could be used for referencing this element from a Resource element.

Parameter

A Parameter element provides the value of a particular algorithm parameter, whose name and format have been specified for the algorithm considered.

```
<!ELEMENT dsig:Parameter ANY>

<!ATTLIST dsig:Parameter
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      type              CDATA       #REQUIRED
>
```

Content Description

ANY:	The contents of a Parameter element consists of ANY valid construct, which is specified on a per algorithm per parameter basis.

Attributes Description

type:	The type of the parameter expressed as a free form string, whose value is specified on a per algorithm basis.

Real

The Real element is a primary data type that is used in the definition of algorithm parameters.

```
<!ELEMENT dsig:Real EMPTY>

<!ATTLIST dsig:Real
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      value             CDATA       #REQUIRED
>
```

Attributes Description

value:	Value of the element given according to the format given below.

Real Format

This specification requires real values to be expressed according to the following pattern:

```
['+'|'-'] n+ ['.' f+] ['E' ('+'|'-') ee]
```

For example, 12, -12.34, +12.34E-01, and +0.5 are valid real values.

RecipientInfo

The RecipientInfo element is used for providing identification and keying material information for the recipient. This element is used either for enabling recognition of a Signature element by a given recipient or when determination of the authentication key consists of the combination of keying material provided by both the recipient and the originator.

Content Description

The content of this element is similar to the one defined for the originator (cf. OriginatorInfo element description).

Resource

The Resource element consists of a unique and unambiguous reference to a resource being authenticated. It is comprised of a resource locator, a fingerprint, and optionally a content-type qualifier.

```
<!ELEMENT dsig:Resource (
      dsig:Locator, dsig:ContentInfo?, dsig:Digest
)>

<!ATTLIST dsig:Resource
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
>
```

Content Description

ContentInfo:	Optional element that provides type information regarding the resource.
Locator:	Locator value that contains a URI [RFC 2396], a fragment identifier, or both. Notice that making use of a fragment identifier for a document content other than XML is out of the scope of this specification and may lead to inconsistent results.
Digest:	Fingerprint of the resource.

Resources

The Resources element consists of a collection of Resource elements. Though inaccessible from the Document element of the Signature DTD, this element is available to more sophisticated constructs that make use of composite documents.

```
<!ELEMENT dsig:Resources ( dsig:Resource+ )>

<!ATTLIST Resources
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      dsig:eval         IDREF       #IMPLIED
      id                ID          #IMPLIED
>
```

Content Description

Resource:	A collection of Resource elements.

Attributes Description

id:	Element identifier that could be used for referencing this element from a Resource element.

Signature

The Signature element constitutes the core of this specification. It is comprised of two sub-elements. The first one is a set of attributes, known as the Manifest, which actually constitutes the authenticated part of the document. The second sub-element consists of the signature value.

```
<!ELEMENT dsig:Signature ( dsig:Manifest, dsig:Value )>

<!ATTLIST dsig:Signature
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      dsig:eval         IDREF       #IMPLIED
      id                ID          #IMPLIED
>
```

Content Description

Manifest:	Element constructed from the set of attributes that constitute the authenticated part of the document.
Value:	The signature value after proper encoding.

Attributes Description

id:	Element identifier that could be used for referencing the Signature element from a Resource element when implementing endorsement.

SignatureAlgorithm

The SignatureAlgorithm element specifies the algorithm to be employed for computation of a signature value.

```
<!ELEMENT dsig:SignatureAlgorithm ( dsig:Algorithm )>

<!ATTLIST dsig:SignatureAlgorithm
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
>
```

Content Description

Algorithm:	Algorithm and parameters to be used for computation of the signature value.

Signatures

The Signatures element consists of a collection of Signature elements. As mentioned in a previous paragraph, this element has been defined for the purpose of facilitating DOM manipulations.

```
<!ELEMENT dsig:Signatures ( dsig:Signature+ )>

<!ATTLIST dsig:Signatures
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
>
```

Content Description

Signature:	A collection of Signature elements.

Value

The Value element consists of a primary data type that is used throughout this proposal for inlining and encoding of arbitrary values.

```
<!ELEMENT dsig:Value ( #PCDATA )>

<!ATTLIST dsig:Value
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      encoding    ( base64 | none ) #IMPLIED    'none'
>
```

Content Description

PCDATA:	Content value after adequate encoding.

Attributes Description

encoding:	This attribute specifies the scheme to be employed for recovering the original byte stream from the content of the element. This specification recognizes the following two schemes: none: the content has not been subject to any particular encoding. This does not preclude

however the use of native XML encoding such as CDATA section or XML escaping.

base64: The content has been encoded by means of the base64 encoding scheme.

Default Document Element

Though it is primarily intended for enabling authentication in other XML applications, the XML Signature DTD specifies a default document element. This definition has been intentionally kept simple and is intended to provide an XML alternative to the ASN1 data types Authenticated Data and Signed Data defined by CMS [x] and PKCS#7 [x] binary syntax standards.

The definition given below addresses the following requirements:

- Authentication of arbitrary contents: This may be done by adequate encapsulation and encoding of the arbitrary contents into the Package element, which shall be further authenticated by means of a Signature element.
- Detached signature: This may be done by means of a Signature element that refers to a resource external to the document.
- Authentication versus signature: The distinction between authentication and signature only depends upon the algorithms being employed for computation of the 'signature' value.
- Plurality of recipients: This consists of the insertion of a plurality of Signature elements, each making use of recipient-dependent keying material.
- Plurality of signers: This consists of the insertion of a plurality of Signature elements, each making use of originator-dependent keying material.

```
<!ELEMENT dsig:Document (
      ( dsig:DigestAlgorithms, dsig:Package+ )?,
      dsig:Signatures, dsig:Certificates?
)>

<!ATTLIST dsig:Document
      xmlns:dsig  CDATA       #FIXED      %xmldsig.dtd;
>
```

Content Description

DigestAlgorithms:	This element has been made mandatory whenever the document embeds the contents to be authenticated. This element specifies the algorithm to be used for computation of the digest of the Package elements, thus enabling one-pass processing.
Package:	This element is used for enveloping and encoding of the contents to be authenticated. Whenever employed, this element shall make use of the dsig:eval global attribute.
Signatures:	This element consists of a collection of Signature elements.
Certificates:	This element consists of a collection of Certificate elements, which may be required by a given key management infrastructure.

Standard Attributes

This specification recognizes the following standard attributes.

Signing-time Attribute

Standard attribute that could be used for specifying the time at which the originator purportedly performed the signature process. This attribute content shall be given as a Date element.

Specification:

URN:	urn:ietf-org:xmldsig-signing-time
Type:	dsig:Date

Example:

```
<dsig:Attribute type='urn:xml-dsig-ietf-org:signing-time'>
    <dsig:Date value='1994-11-05T16:15:02.031-0500'\>
</dsig:Attribute>
```

Digest Algorithms

This specification contemplates two types of digest algorithms:

- Surface string digest algorithms: These algorithms do not have any particular knowledge about the content being digested and operate on the raw content value. Changes in the surface string of a given content affect directly the value of the digest being produced.
- Canonical digest algorithms: These algorithms have been tailored for a particular content type and produce a digest value that depends upon the core semantics of such content. Changes limited to the surface string of a given content do not affect the value of the digest being produced.

SHA1

Surface string digest algorithm designed by NIST and NSA for use with the Digital Signature Standard. This algorithm is documented in NIST FIPS Publication 180-1.

Specifications

URN:	urn:nist-gov:sha1

Parameters:

This algorithm does not require any parameter.

DOM-HASH

XML canonical digest algorithm proposed by IBM Tokyo Research Laboratory and documented in the DOMHASH proposal [x]. This algorithm operates on the DOM representation of the document and provides an unambiguous means for recursive computation of the hash value of the nodes that constitute the DOM tree. This algorithm has many applications such as computation of digital signature and synchronization of DOM trees. However, because the hash value of an element is computed from the hash values of the inner elements, this algorithm is better adapted to small documents that do not require one-pass processing.

As of today, this algorithm is limited to the contents of an XML document and, therefore, does not provide for authentication of the internal or external subset of the DTD.

The DOM-HASH algorithm requires a single parameter, which shall consist of a surface string digest algorithm such as SHA1.

Specifications

URN:	urn:ibm-com:dom-hash

Parameters:

digest-algorithm	Surface string digest algorithm to be used for computation of the digest value..
type:	dsig:Algorithm
default:	urn:nist-gov:sha1

Example

```
<dsig:Algorithm type='urn:ibm-com:dom-hash'>
    <dsig:Parameter type='digest-algorithm'>
        <dsig:Algorithm type='urn:nist-gov:sha1'/>
    </dsig:Parameter>
</dsig:Algorithm>
```

XHASH

XML canonical digest algorithm proposed by GlobeSet and documented in the XHASH proposal [x]. This algorithm has been inspired by the DOM-HASH proposal, but operates closer to the surface string of the document. Elements and attributes are subject to formalization in a way quite similar to the one proposed by DOM-HASH - XML delimiters are represented by binary values and entities are replaced by their actual values. However, formalization happens as elements are acquired. Furthermore, this algorithm takes into account some specifics of this specification (e.g. dsig:eval attribute).

The XHASH algorithm makes use of two parameters. The first one consists of a surface string digest algorithm such as SHA1. The second one, optional, may be used for specifying how non-significant SPACE characters shall be handled by default. Actually, the XML Specifications define the xml:space attribute that could be used for specifying if non-significant SPACE characters are to be preserved. However, possible values for this attribute are limited to 'default' and 'preserve'. Thus, there is no known way to explicitly specify that non-significant SPACE characters should be discarded.

Specifications

URN:	urn:globeset-com:xhash

Parameters:

digest-algorithm	Surface string digest algorithm to be used for computation of the digest value.
type:	dsig:Algorithm
default:	urn:nist-gov:sha1
white-spaces	Default processing of non-significant SPACE characters.
type:	dsig:Keyword
value:	preserve: Non-significant SPACE characters are to be preserved in a way similar to what should be done in presence of an xml:space preserve attribute.
	ignore: Unless overridden by means of an xml:space preserve attribute, non-significant SPACE characters shall be ignored during computation of the canonical form of the contents.
default:	ignore

Example

```
<dsig:Algorithm type='urn:globeset-com:xhash'>
    <dsig:Parameter type='digest-algorithm'>
        <dsig:Algorithm type='urn:nist-gov:sha1'/>
    </dsig:Parameter>
    <dsig:Parameter type='white-spaces'>
        <dsig:Keyword value='ignore'/>
    </dsig:Parameter>
</dsig:Algorithm>
```

Key Agreement Algorithms

A key-agreement algorithm consists of a function that is used for deriving a one-time session key from a given master key. Usage of one-time session keys prevents some kinds of attacks that require a large volume of cipher-text to be produced with a given key.

Usage of a key-agreement algorithm is recommended when authentication is based upon a shared secret. This shared secret could have been exchanged either by offline means (e.g. mail) or computed dynamically by means of a key-exchange algorithm such as Diffie Hellman [x].

PKCS12-PBE

Key-agreement algorithm proposed by RSA Laboratories and documented in PKCS12 [x]. This algorithm is a generalization of the PBE algorithm defined in PKCS5 [x] and provides for the generation of symmetric keys and other cryptographic parameters from an established password.

This algorithm requires three parameters. The first one consists of a one-way hash function (i.e. SHA1), the second one of a random string (salt), and the last one of an iteration count.

Specifications

URN:	urn:rsasdi-com:pkcs12-pbe

Parameters:

digest-algorithm	One-way hash function used as the pseudo-random number generator.
type:	dsig:Algorithm
default:	urn:nist-gov:sha1
random-string	Random string value used to seed the PRNG.
type:	dsig:Value
default:	no default.
iteration-count	
type:	dsig:Integer
default:	256

Example

```
<dsig:Algorithm type='urn:rsasdi-com:pkcs12-pbe'>
    <dsig:Parameter type='digest-algorithm'>
        <dsig:Algorithm type='urn:nist-gov:sha1'/>
    </dsig:Parameter>
    <dsig:Parameter type='random-string'>
        <dsig:Value encoding='base64'>
            Abkirjegks123qwgtawd456g47
        </dsig:Value>
    </dsig:Parameter>
    <dsig:Parameter type='iteration-count'>
        <dsig:Integer value='128'/>
    </dsig:Parameter>
```

```
</dsig:Algorithm>
```

Key Exchange Algorithms

A key-exchange algorithm enables dynamic establishment of a master secret key that results from the combination of keying material provided by the parties involved in an exchange. The parties may further establish a one-time session key from such a master secret key by means of a key-agreement algorithm.

Key-exchange algorithms shall not be defined in the body of a signed document. Their usage is implicit and depends solely upon the keying material being used for authentication.

Diffie Hellman

Key-exchange algorithm named from its authors and documented in X9.42.

Signature Algorithms

This specification abusively uses the terminology of 'digital signature' for qualifying indifferently digital signature and message authentication codes. Thus, the signature algorithms contemplated herein include public-key digital signature algorithms such as DSA and message authentication codes such as HMAC.

HMAC

Message Authentication Code proposed by H. Krawczyk and al. and documented in RFC2104.

This specification adopts a scheme that differs a bit from the common usage of this algorithm – computation of the MAC is performed on the hash of the contents being authenticated instead of the actual contents. Thence, the actual signature value output by the algorithm might be depicted as follows:

```
SignatureValue = HMAC( SecretKet, H(dsig:Manifest))
```

This specification also considered HMAC output truncation such as proposed by Preneel and van Oorschot. In their paper [x] these two researchers have shown some analytical advantages of truncating the output of hash-based

MAC functions. Such output truncation is also considered in the RFC document.

HMAC requires three parameters. The first one consists of a canonical digest algorithm. The second one consists of a hash function. The last one is optional and specifies the length in bit of the truncated output. If this last parameter is absent, no truncation shall occur.

Specifications

URN:	urn:ietf-org:hmac

Parameters:

digest-algorithm	Canonical or surface-string digest algorithm to be is used for computation of the Manifest fingerprint.
type:	dsig:Algorithm
default:	urn:nist-gov:sha1
hash-function	Hash function that is used to compute the MAC value from the secret key and the fingerprint of the signature Manifest.
type:	dsig:Algorithm
default:	urn:nist-gov:sha1
output-length	Length in bits of the truncated MAC value.
type:	dsig:Integer
default:	no default.

Signature Value Encoding:

The output of this algorithm can be assumed as a large integer value. The signature value shall consist of the octet-encoded value of this integer. Integer to octet-stream conversion shall be done according to PKCS#1 [x] specification with a k parameter equals to ((Hlen +7) mod 8), Mlen being the length in bits of the MAC value.

Example

```
<dsig:Algorithm type='urn:ietf-org:hmac'>
    <dsig:Parameter type='digest-algorithm'>
        <dsig:Algorithm type='urn:globeset-com:xhash'/>
    </dsig:Parameter>
    <dsig:Parameter type='hash-function'>
        <dsig:Algorithm type='urn:nist-gov:sha1'/>
    </dsig:Parameter>
    <dsig:Parameter type=output-length'>
        <dsig:Integer value='128'/>
    </dsig:Parameter>
</dsig:Algorithm>
```

DSA

Public-key signature algorithm proposed by NIST for use with the Digital Signature Standard. This standard is documented in NIST FIPS Publication 186 [x] and ANSI X9.30 [x].

The DSA algorithm requires a single parameter, which consists of the canonical digest algorithm to be used for computing the fingerprint of the signature Manifest.

Specifications

URN:	urn:nist-gov:dsa

Parameters:

digest-algorithm	Canonical or surface-string digest algorithm to be is used for computation of the Manifest fingerprint.
type:	dsig:Algorithm
default:	urn:nist-gov:sha1

Signature Value Encoding:

The output of this algorithm consists of a pair of integers usually referred by the pair (r, s). The signature value shall consist of the concatenation of two octet-streams that respectively result from the octet-encoding of the values r and s. Integer to octet-stream conversion shall be done according to PKCS#1 [x] specification with a k parameter equals to 20.

Example

```
<dsig:Algorithm type='urn:nist-gov:dsa'>
    <dsig:Parameter type='digest-algorithm'>
        <dsig:Algorithm type='urn:globeset-com:xhash'/>
    </dsig:Parameter>
```

```
</dsig:Algorithm>
```

RSA-Encryption

Public-key signature algorithm proposed by RSA Laboratories and documented in PKCS#1 [x].

This specification adopts the RSA encryption algorithm with padding block type 01. For computing the signature value, the signer shall first digest the signature Manifest and then encrypt the resulting digest with his private key.

This signature algorithm requires a single parameter, which consists of the canonical digest algorithm to be used for computing the fingerprint of the signature Manifest.

Specifications

URN:	urn:rsasdi-com:rsa-encryption

Parameters:

digest-algorithm	Canonical or surface-string digest algorithm to be is used for computation of the Manifest fingerprint.
type:	dsig:Algorithm
default:	urn:nist-gov:sha1

Signature Value Encoding:

The output of this algorithm consists of single octet-stream. No further encoding is required.

Example

```
<dsig:Algorithm type='urn:rsasdi-com:rsa-encryption'>
    <dsig:Parameter type='digest-algorithm'>
        <dsig:Algorithm type='urn:globeset-com:xhash'/>
    </dsig:Parameter>
</dsig:Algorithm>
```

ECDSA

Public-key signature algorithm proposed independently by Neil Koblitz and Victor Miller. This algorithm is being proposed as an ANSI standard and is documented in ANSI X9.62 standard proposal [x] and IEEE/P1363 standard draft proposal [x].

The ECDSA algorithm requires a single parameter, which consists of the canonical digest algorithm to be used for computing the fingerprint of the signature Manifest.

Specifications

URN:	urn:ansi-org:ecdsa

Parameters:

digest-algorithm	Canonical or surface-string digest algorithm to be is used for computation of the Manifest fingerprint.
type:	dsig:Algorithm
default:	urn:nist-gov:sha1

Signature Value Encoding:

The output of this algorithm consists of a pair of integers usually referred by the pair (r, s). The signature value shall consist of the concatenation of two octet-streams that respectively result from the octet-encoding of the values r and s. Integer to octet-stream conversion shall be done according to PKCS#1 [x] specification with a k parameter equals to 20.

Example

```
<dsig:Algorithm type='urn:ansi-org:ecdsa'>
    <dsig:Parameter type='digest-algorithm'>
        <dsig:Algorithm type='urn:globeset-com:xhash'/>
    </dsig:Parameter>
</dsig:Algorithm>
```

References

[...more to come…]

Signature DTD

```
<-- XML Signature DTD – 990404 – Revision 0 --->

<!ENTITY % xmldsig.dtd 'http://www.dtd.reg.int/xmldsig'>

<!ELEMENT dsig:Document (
      ( dsig:DigestAlgorithms, dsig:Package+ )?,
      dsig:Signatures, dsig:Certificates?
)>

<!ATTLIST dsig:Document
      xmlns:dsig  CDATA       #FIXED      %xmldsig.dtd;
>

<!ELEMENT dsig:Algorithm ( dsig:Parameter* )>

<!ATTLIST dsig:Algorithm
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      id                ID          #IMPLIED
      type              CDATA       #REQUIRED
>

<!ELEMENT dsig:Attribute ANY>

<!ATTLIST dsig:Attribute
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      type              NMTOKEN     #REQUIRED
      critical    ( true | false )  #IMPLIED    'false'
>

<!ELEMENT dsig:Attributes ( dsig:Attribute+ )>

<!ATTLIST dsig:Attributes
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
>

<!ELEMENT dsig:Certificate (
      dsig:IssuerAndSerialNumber,
      ( dsig:Value | dsig:Locator )
)>

<!ATTLIST dsig:Certificate
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      type              NMTOKEN     #REQUIRED
>

<!ELEMENT dsig:Certificates ( dsig:Certificate+ )>

<!ATTLIST dsig:Certificates
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
>

<!ELEMENT dsig:ContentInfo EMPTY>
```

```
<!ATTLIST dsig:ContentInfo
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      type              CDATA       #REQUIRED
      subtype           CDATA       #IMPLIED
>

<!ELEMENT dsig:Date EMPTY>

<!ATTLIST dsig:Date
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      value             CDATA       #REQUIRED
>

<!ELEMENT dsig:Digest ( dsig:Algorithm, dsig:Value )>

<!ATTLIST dsig:Digest
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
>

<!ELEMENT dsig:DigestAlgorithms ( dsig:Algorithm+ )>

<!ATTLIST dsig:DigestAlgorithms
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
>

<!ELEMENT dsig:Identifier EMPTY>

<!ATTLIST dsig:Identifier
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      value             CDATA       #REQUIRED
>

<!ELEMENT dsig:Integer EMPTY>

<!ATTLIST dsig:Integer
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      value             CDATA       #REQUIRED
>

<!ELEMENT dsig:IssuerAndSerialNumber EMPTY>

<!ATTLIST dsig:IssuerAndSerialNumber
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      issuer            CDATA       #REQUIRED
      number            CDATA       #REQUIRED
>

<!ELEMENT dsig:KeyAgreementAlgorithm ( dsig:Algorithm )>

<!ATTLIST dsig: KeyAgreementAlgorithm
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
>

<!ELEMENT dsig:Keyword EMPTY>
```

```
<!ATTLIST dsig:Keyword
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      value             CDATA       #REQUIRED
>

<!ELEMENT dsig:Locator EMPTY>

<!ATTLIST dsig:Locator
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      xml:link          CDATA       #FIXED      ’simple’
      href              CDATA       #REQUIRED
>

<!ELEMENT dsig:Manifest (
      dsig:Resources, dsig:Attributes?,
      dsig:OriginatorInfo, dsig:RecipientInfo?,
      dsig:KeyAgreementAlgorithm?, dsig:SignatureAlgorithm
)>

<!ATTLIST dsig:Manifest
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
>

<!ELEMENT dsig:OriginatorInfo ANY>

<!ATTLIST dsig:OriginatorInfo
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
>

<!ELEMENT dsig:RecipientInfo ANY>

<!ATTLIST dsig: RecipientInfo
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
>

<!ELEMENT dsig:Package (
      dsig:ContentInfo?, dsig:Value
)>

<!ATTLIST dsig:Package
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      dsig:eval         IDREF       #IMPLIED
      id                ID          #IMPLIED
>

<!ELEMENT dsig:Parameter ANY>

<!ATTLIST dsig:Parameter
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      type              CDATA       #REQUIRED
>

<!ELEMENT dsig:Real EMPTY>

<!ATTLIST dsig:Real
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      value             CDATA       #REQUIRED
>

<!ELEMENT dsig:Resource (
      dsig:Locator, dsig:ContentInfo?, dsig:Digest
)>

<!ATTLIST dsig:Resource
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
>

<!ELEMENT dsig:Resources ( dsig:Resource+ )>

<!ATTLIST Resources
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      dsig:eval         IDREF       #IMPLIED
      id                ID          #IMPLIED
>

<!ELEMENT dsig:Signature ( dsig:Manifest, dsig:Value )>

<!ATTLIST dsig:Signature
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      dsig:eval         IDREF       #IMPLIED
      id                ID          #IMPLIED
>

<!ELEMENT dsig:SignatureAlgorithm ( dsig:Algorithm )>

<!ATTLIST dsig:SignatureAlgorithm
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
>

<!ELEMENT dsig:Signatures ( dsig:Signature+ )>

<!ATTLIST dsig:Signatures
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
>

<!ELEMENT dsig:Value ( #PCDATA )>

<!ATTLIST dsig:Value
      xmlns:dsig        CDATA       #FIXED      %xmldsig.dtd;
      encoding    ( base64 | none ) #IMPLIED    'none'
>
```

Embedded Content Example

This example illustrates use of the default document element for attachment and authentication of an arbitrary piece of information.

```
<?xml version='1.0'?>
<!DOCTYPE dsig:Document PUBLIC 'urn:ietf-org:xmldsig.dtd'
  SYSTEM 'http://www.dtd.reg.int/dtd/xmldsig.dtd'>

<dsig:Document>

  <dsig:DigestAlgorithms>
    <dsig:Algorithm id='xhash' type='urn:com-globeset:xhash'/>
  </dsig:DigestAlgorithms>

  <dsig:Package id='data' dsig:eval='xhash'>
    <dsig:ContentInfo
      type='urn:mime:application%2fmsword'/>
    <dsig:Value>
      abncjflf311257gghn6mj2k134h64AANHdd12==
    </dsig:Value>
  </dsig:Package>

  <dsig:Signatures>
    <dsig:Signature>
      <dsig:Manifest>

        <dsig:Resources>
          <dsig:Resource>
            <dsig:Locator href='data'/>
            <dsig:Digest>
              <dsig:Algorithm type='urn:com-globeset:xhash'/>
              <dsig:Value encoding='base64'>
                bndWGryrt245u6t1dgURTIrr4ir5=
              </dsig:Value>
            </dsig:Digest>
          </dsig:Resource>
        </dsig:Resources>

        <dsig:Attributes>
          <dsig:Attribute
            type='urn:ietf-org:xmldsig-signing-time'>
            <dsig:Date value='1998-10-29T13:26-0500'/>
          </dsig:Attribute>
        </dsig:Attributes>

        <dsig:OriginatorInfo>
          <dsig:IssuerAndSerialNumber
                issuer='o=GlobeSet Inc., c=US'
                number='123456789102356'/>
        </dsig:OriginatorInfo>

        <dsig:SignatureAlgorithm>
          <dsig:Algorithm type='urn:rsasdi-com:rsa-encryption'>
            <dsig:Parameter type='digest-algorithm'>
              <dsig:Algorithm type='urn:globeset-com:xhash'/>
            </dsig:Parameter>
          </dsig:Algorithm>
        </dsig:SignatureAlgorithm>

      </dsig:Manifest>
      <dsig:Value>
        xsqsfasDys2h44u4ehJDe54he5j4dJYTJ=
      </dsig:Value>

    </dsig:Signature>

  </dsig:Signatures>

  <dsig:Certificates>

    <dsig:Certificate type='urn:X500:X509v3'>
      <dsig:IssuerAndSerialNumber
            issuer='o=GlobeSet Inc., c=US'
            number='123456789102356'/>
      <dsig:Locator href='http://certs.globeset.com/smith.der'/>
    </dsig:Certificate>

    <dsig:Certificate type='urn:X500:X509v3'>
      <dsig:IssuerAndSerialNumber
            issuer='o=GlobeSet Inc., c=US'
            number='123456789102356'/>
      <dsig:Value>
        xsqsfasDys2h44u4ehJDe54he5j4dJYTJ=
      </dsig:Value>
    </dsig:Certificate>

  </dsig:Certificates>

</dsig:Document>
```

Detached Signature Example

This example illustrates use of the default document element for production of a detached-signature. This example assumes that recipient and originator benefit from a prior relationship.

```
<?xml version='1.0'?>
<!DOCTYPE dsig:Document PUBLIC 'urn:ietf-org:xmldsig.dtd'
  SYSTEM 'http://www.dtd.reg.int/dtd/xmldsig.dtd'>

<dsig:Document>

  <dsig:Signatures>

    <dsig:Signature>
      <dsig:Manifest>

        <dsig:Resources>
          <dsig:Resource>
```

```
            <dsig:Locator ref='http://www.globeset.com/xml.doc'/>
            <dsig:ContentInfo
              type='urn:mime:application%2fmsword'/>
            <dsig:Digest>
              <dsig:Algorithm type='urn:nist-gov:sha1'/>
              <dsig:Value>
                bndWGryrt245u6t1dgURTIrr4ir5=
              </dsig:Value>
            </dsig:Digest>
          </dsig:Resource>
        </dsig:Resources>

        <dsig:OriginatorInfo>
          <dsig:Identifier value="0695123"/>
        </dsig:OriginatorInfo>

        <dsig:SignatureAlgorithm>
          <dsig:Algorithm type='urn:ietf-org:hmac'>
            <dsig:Parameter type='digest-algorithm'>
              <dsig:Algorithm type='urn:globeset-com:xhash'/>
            </dsig:Parameter>
            <dsig:Parameter type='output-length'>
              <dsig:Integer value='128'/>
            </dsig:Parameter>
        </dsig:SignatureAlgorithm>

      </dsig:Manifest>

      <dsig:Value>
        xsqsfasDys2h44u4ehJDe54he5j4dJYTJ=
      </dsig:Value>

    </dsig:Signature>

  </dsig:Signatures>

</dsig:Document>
```

Domain-specific Example

This example illustrates how to leverage the XML Signature DTD to enable authentication in another XML application.

This application contemplates the production of authenticated Ticket documents that conform to the following DTD

```
<!-- Include XML DSIG Definitions -->
  <!ENTITY % XmlDsigDtd SYSTEM
    'http://www.dtd.reg.int/dtd/xmldsig.dtd'>
  %XmlDsigDtd;
  <!-- Application Specific Definitions -->
  <!ELEMENT Ticket (
    Body, dsig:Signature, dsig:Certificate
  )>

  <!ELEMENT Body (Event, Beneficiary)>
  <!ATTLIST Body
    number      ID          #REQUIRED
  >

  <!ELEMENT Event EMPTY>
  <!ATTLIST Event
    desc        CDATA       #REQUIRED
    date        CDATA       #REQUIRED
  >

  <!ELEMENT Beneficiary EMPTY>
  <!ATTLIST Beneficiary
    name        CDATA       #REQUIRED
    uid         CDATA       #REQUIRED
  >
```

The following consists of a Ticket document that conforms to the previous DTD. The system makes use of a public-key signature algorithm (RSA) and relies upon X509v3 credentials.

```
<?xml version='1.0'?>
<!DOCTYPE Ticket>

<Ticket>

  <Body number='120456789'>
    <Event
      desc='concert in Austin'
      date='1999-04-12T20:30-0500'/>
    <Beneficiary
      name='John smith'
      ssno='435-56-4023'/>
   </Body>

  <dsig:Signature>
    <dsig:Manifest>

      <dsig:Resources>
        <dsig:Resource>
          <dsig:Locator href='120456789'/>
          <dsig:Digest>
            <dsig:Algorithm type='urn:com-globeset:xhash'/>
            <dsig:Value encoding='base64'>
              bndWGryrt245u6t1dgURTIrr4ir5=
            </dsig:Value>
          </dsig:Digest>
        </dsig:Resource>
      </dsig:Resources>

      <dsig:OriginatorInfo>
        <dsig:IssuerAndSerialNumber
          issuer='o=GlobeSet Inc., c=US'
          number='123456789102356'/>
      </dsig:OriginatorInfo>

      <dsig:SignatureAlgorithm>
        <dsig:Algorithm type='urn:rsasdi-com:rsa-encryption'>
          <dsig:Parameter type='digest-algorithm'>
            <dsig:Algorithm type='urn:globeset-com:xhash'/>
          </dsig:Parameter>
        </dsig:Algorithm>
      </dsig:SignatureAlgorithm>

    </dsig:Manifest>

    <dsig:Value>
      xsqsfasDys2h44u4ehJDe54he5j4dJYTJ=
    </dsig:Value>

  </dsig:Signature>

  <dsig:Certificate type='urn:X500:X509v3'>
    <dsig:IssuerAndSerialNumber
      issuer='o=GlobeSet Inc., c=US'
      number='123456789102356'/>
    <dsig:Value>
      xsqsfasDys2h44u4ehJDe54he5j4dJYTJ=
    </dsig:Value>
  </dsig:Certificate>

</dsig:Ticket>
```

Comments and Suggestions

This chapter is a temporary repository for all the comments and suggestions that have been received in regard to this specification. It constitutes the primary source of information for forthcoming revisions of this specification.

#98072901 – Syntax too verbose

Origin:	General
Status:	Superceded by 98091001, 98091002

Comments:

"The syntax is far too much verbose."

Author Notes:

There are multiple optimizations that could be envisioned:

- Promote syntax compactness instead of element reusability.
- Adopt syntax that enables shared algorithm definitions.
- Adopt default attribute and parameter values.

#98091001 – Enable shared algorithm definitions

Origin:	Richard D. Brown
Status:	Opened

Comments:

"Considering that in most circumstances the digest of authenticated resources will be computed by means of a common digest algorithm, it seems preferable to define a syntax that allows shared algorithm definitions.

Several approaches might be considered. The first one consists of adding a DigestAlgorithms element in every signature Manifest and using some linking mechanism (i.e. IDREF) to bind a digest value with a digest algorithm. Another approach is to allow definitions throughout the document and requiring the canonicalizer to inline the algorithm definition in the Digest elements."

Author Notes:

The first approach does not prevent replication of algorithm definitions in the header of the document and the different signature Manifests. On the other hand, the second approach requires particular behaviors in the canonicalizers and therefore cannot suffice with a surface string digest algorithm.

#98091002 – Promote compactness over reusability

Origin:	Richard D. Brown
Status:	Opened

Comments:

"The current specification makes use of reusable element types. This approach obviously increases the "verbosity" of the syntax. It might be preferable to promote compactness instead of reusability."

For example, the following optimizations might be considered:

- Collapsing Locator into Resource.
- Collapsing ContentInfo into Package.
- Replacing basic data types (i.e. Integer, Date, etc…) by a DCD attribute and a PCDATA contents on the parent element.

Author Notes:

#98111301 – Add a Resource criticality attribute

Origin:	Milton M. Anderson
Status:	Opened

Comments:

"The definition of <resource> does not contain a feature to allow the signer to declare that the resource is critical to the validity of the signature."

Author Notes:

I do not have all the elements for judging how relevant is this particular comment, but my first guess is that a signature applies to a particular content and semantics verification shall be left to the application layer.

#98111302 – Remove dsig:eval global attribute

Origin:	Milton M. Anderson
Status:	Closed

Comments:

"Having the dsig:eval attribute in the element that defines the authenticated block is probably a bad idea, since different signers can use different hash algorithm."

Author Notes:

Others have made a similar comment, but we have reached a different conclusion: dsig:eval shall be an IDREFS instead of an IDREF.

#98111303 – Add a logging attribute

Origin:	Milton M. Anderson
Status:	Closed

Comments:

"No ability to mark data for logging by a signing hardware is included."

Author Notes:

Also, very much application specific. I am not quite sure this shall be a concern for the XML DSIG Proposal.

#98111304 – Add a signature purpose attribute

Origin:	Milton M. Anderson
Status:	Closed

Comments:

"The signature doesn’t have a "type of signature" element."

Author Notes:

This may be provided at the application level by means of a complementary attribute.

#98111305 – Add a Nonce in the Manifest

Origin:	Milton M. Anderson
Status:	Closed

Comments:

"No nonce to be used in hashing the resource is defined."

Author Notes:

Milton refers to a scheme that could be used for preventing birthday-attacks against the digest algorithm.

E-Check makes use of this artifice to make sure that a fraudulent signer or a Trojan Horse on the signer’s computer does not have full control over the data being signed by the signing device. If the attacker were able to find two messages M1 and M2 such that H(M1) = H(M2), it could send M1 to the device and then substitute M2 later. The hardware log will record that M1 has been signed and not M2.

This problem may be addressed in the hardware log by itself. The device may sign H(M1), pick a nonce at random, and log the nonce value along with H(nonce||M1) for example.

#98112501 – Leverage DCD and other specifications

Origin:	Hiroshi Maruyama
Status:	Opened

Comments:

"For data types, I think we could refer to other activities such as DCD."

Author Notes:

Agreed as long as these specifications are adopted in time. Notice that DCD provides mostly for basic data types. This will not resolve the problem for the constructed data types such as IssuerAndSerialNumber, Package, etc…

#98121501 – IssuerAndSerialNumber is too restrictive

Origin:	Richard D. Brown

Status:	Opened

Comments:

"Identifying a certificate by means of the constructed data types IssuerAnd-SerialNumber might be too restrictive. It is not obvious that this construct is sufficient for certificates other than X509v3. The specification shall adopt syntax a bit more opened.

For example, a certificate might be uniquely identified by means of a Identifier element, whose syntax depends upon the type of the certificate."

Author Notes:

#98122601 – Allow multiple Resource in Manifest

Origin:	Don Eastlake
Status:	Adopted (990404)

Comments:

"I believe that multiple occurrences of Resource should be permitted in the Manifest"

Author Notes:

The Manifest now requires a Resources element.

#98122602 – Add a base locator for HREFs

Origin:	Don Eastlake – IOTP WG
Status:	Opened

Comments:

"Some of the IOTP concerns about many huge locator HREFs could be satisfied if a "base" attribute were permitted at the Resources level or, even more general, a Base element, which effected all following resource."

Author Notes:

#98122603 – Rename the attribute dsig:eval

Origin:	Don Eastlake
Status:	Opened

Comments:

"I do not like the choice of "eval" even if it is arbitrary. It makes me think of Lisp or the like. I would expect it to evaluate arithmetic expressions or the like. I think the previous "hash" was better and perhaps "dsig:digest" would be best…"

Author Notes:

#98122604 – Default encoding attribute to base64

Origin:	Don Eastlake
Status:	Opened

Comments:

"What's about defaulting to base64 instead of none for the encoding."

Author Notes:

#99010201 – Add a ID attribute to Algorithm

Origin:	Mark Linehan
Status:	Superceded

Comments:

"I suggest adding an "id" attribute to the Algorithm element, and adding an algref attribute to any element that contains an Algorithm. Purpose is to avoid repeating the same Algorithm text in many places."

Author Notes:

Something like has been proposed by the IOTP WG and is already referenced.

#99010202 – Provide a default digest algorithm

Origin:	Mark Linehan
Status:	Closed

Comments:

"I suggest that it would be helpful to have a way to declare a default or standard digest algorithm. Reason: in most cases, the same algorithm will be used throughout a document."

Author Notes:

Adequate optimization should enable use of shared definitions. In such circumstances, the overhead on a Resource element will be limited to an IDREF. At first, removing all algorithm references from the Resource element does not seem a good idea.

#99010203 – Add a type to RecipientInfo and OriginatorInfo

Origin:	Mark Linehan
Status:	Closed

Comments:

"I suggest that OriginatorInfo and RecipientInfo have a "type" attribute for the same reason as the Attribute element: to identify the format of the ANY content."

Author Notes:

RecipientInfo and OriginatorInfo are expected to be a collection of "attributes". Therefore, it does not make sense to define a "type" attribute for these elements.

#99020301 – Adopt URL instead of URI

Origin:	Joseph Reagle
Status:	Opened

Comments:

"What's about adopting URLs instead of URNs. This will prevent registration requirements. At the least, the specification should allow URIs."

Author Notes:

#99021201 – Allow multiple signatures on a Manifest

Origin:	IOTP WG
Status:	Opened

Comments:

"Ability to have multiple signatures on a single Manifest and ability to adhere a recipient to a Signature."

To address these concerns, IOTP DTD proposes the following construct:

```
<!-- DTD Extract -->

<!ELEMENT Signature (Manifest, Value+)>
<!ELEMENT Manifest (
  Resources, Attributes?, OriginatorInfo, RecipientInfo+
)>
<!ELEMENT RecipientInfo ANY>
<!ATTLIST RecipientInfo
    SignatureValueRef   IDREF   #REQUIRED
    SignatureCertRef    IDREF   #IMPLIED
>
```

```
<!-- Signature Example -->

<Signature>
  <Manifest>
    <Resources>
      ...
    </Resources>
    <OriginatorInfo>
      ...
    </OriginatorInfo>
    <RecipientInfo
      SignatureValueRef=’sigv1’
      SignatureCertRef=’cert1’>
      ...
    </RecipientInfo>
    <RecipientInfo
      SignatureValueRef=’sigv2’
      SignatureCertRef=’cert2’>
      ...
    </RecipientInfo>
      ...
  </Manifest>
  <Value id=’sigv1’>
    aBcdsejhtksagnbf==
  </Value>
  <Value id=’sigv2’>
    ehlekjrekkjrk==
  </Value>
</Signature>

<Certificates>
  <Certificate id=’cert1’ ...>
    ...
  </Certificate>
  <Certificate id=’cert2’ ...>
    ...
  </Certificate>
</Certificate>
```

Author Notes:

Assuming that the benefit expected from this construct is to prevent replication of a large Manifest in multiple Signature elements, I would remind that this specification allows for shared Resources element.

I can see at least three potential drawbacks with this construct:

- Privacy: A signature value cannot be verified unless all the RecipientInfo elements are preserved into the Manifest. In some circumstances, it may not be desirable to disclose the identity of the other recipients. Notice however that this construct does not preclude the creation of independent signature elements.

- Complexity: This construct is obviously more complicated than the one currently proposed. Dual forward references are not always easy to handle.
- Inconsistency: Applying multiple signatures to a single document usually implies that the application has adopted some secret-key authentication scheme. In such circumstances, the originator may be known by the recipients under different names or accounts. But, this construct does not allow replication of the OriginatorInfo element (per recipient basis), which is supposed to carry such pieces of information.

#99021202 – Enable shared digest algorithm definitions

Origin:	IOTP WG
Status:	Superceded by 98091002

Comments:

"Ability to share digest algorithms for signatures, digest, and canonicalization"

To address this concern, the IOTP DTD proposes to insert a collection of algorithm definitions into the signature Manifest and to use a linking mechanism for binding Resource definitions and signature algorithms to these shared definitions.

```
<!-- DTD Extract -->

<!ELEMENT Manifest (
    Algorithm+,
    Resource+,
    Attributes?,
    OriginatorInfo,
    RecipientInfo+
)>

<!ELEMENT Digest (#PCDATA)>
<!ATTLIST Digest
    DigestAlgorithmRef  IDREF   #REQUIRED
>
```

Author Notes:

Potential solution to optimizing the syntax. However, I would suggest replacing the collection of Algorithm elements by a DigestAlgorithms element. Also, I would limit this functionality to the Digest elements.

#99031601 – Remove criticality attribute on Attribute

Origin:	Dave Solo, Eric Riscola
Status:	Opened

Comments:

… IETF Meeting – following a presentation of criticality flag …

Dave: "it's a bad idea: experience in DMS was to remove it"

Eric: "I reinforce dave: this bogs down every new attribute with a debate over 'criticality' … PKIX and S/MIME show signs of precisely this kind of bloat."

Author Notes:

Tend to agree that criticality shall be a matter of the application layer and, therefore, criticality attribute is not necessary in the syntax.

#99040101 – Remove Attributes element

Origin:	Yoshiaki Kawatsura
Status:	Closed

Comments:

"I do not understand why the Attributes element is needed."

Author Notes:

Collection elements such Attributes, Certificates, and Signatures has been proposed to facilitate DOM manipulation. For example, one may call some trust verification engine by passing the Certificates sub-tree. This construct enables containment of similar and related elements.

#99040102 – Allow attributes on Resource

Origin:	Richard D. Brown
Status:	Opened

Comments:

"It seems that allowing per Resource attributes may have interesting applications. For example, a rating standard could define a rating:Public attribute that could be associated directly with the Resource element. In such circumstances, a rating standard could almost suffice with the current DTD definitions."

Author Notes:

#99040103 – Add an CanonicalizerAlgorithm element

Origin:	Richard D. Brown
Status:	Opened

Comments:

"All signature schemes require canonicalization and digest of the Manifest. Thence, all the signature algorithms require at least a digest-algorithm parameter and this has lead to some inconsistencies such as requiring a digest algorithm for rsa-encryption or two hash functions for HMAC.

It may be preferable to add a CanonicalAlgorithm element in the signature Manifest. This element will identify the digest/canonical algorithm to be used for computation of the fingerprint of the Signature Manifest."

```
<!ELEMENT Manifest(
    DigestAlgorithms,
    Resources,
    Attributes?,
    OriginatorInfo,
    RecipientInfo,
    KeyAgreementAlgorithm?,
    CanonicalizerAlgorithm,
    SignatureAlgorithm
)>
```

Author Notes:

#99040104 – Allow attributes on Package

Origin:	Richard D. Brown
Status:	Opened

Comments:

"Similarly to adding attributes to the Resource element. For example, such added functionality could be used for attaching an origin attribute to a package. This may be used for binding indirectly a detached-signature with an internal Package element."

Author Notes:

#99040105 – Change Attributes contents to ANY

Origin:	Richard D. Brown
Status:	Opened

Comments:

"Currently, the Attributes element consists of a collection of Attribute elements, which specify a type and contain an inner value element.

An alternative a bit more opened would consist of defining Attributes as an element of ANY contents and use constructed attribute element."

```
<Resource href='http://www.w3c.org/doc.xml'>
  <Attributes>
    <rating:Audience value='all'/>
  </Attributes>
  ...
</Resource>
```

Author Notes:

One could argue that origin, digest, and the forth are also attributes of the resource and, therefore, could be sealed in the Attributes element. In fact, we could remove the Attributes element and define Resource as an element of ANY contents. But, if we were to do so, it would be impossible to enforce the presence of mandatory attribute elements such as resource locator and digest by means of the DTD.

If this suggestion were to be adopted, we may consider converting the ContentInfo element into a standard attribute.

#99040106 – Change ContentInfo contents to PCDATA

Origin:	Richard D. Brown
Status:	Opened

Comments:

"The specification proposes a ContentInfo element with a type and subtype attribute. The type attribute is defined as an URN. Unfortunately, URN specification does not allow the character "/" in the NSS. Thence, it is not possible to map directly existing MIME types into an URN without adopting adequate NSS encoding. For example, "application/msword" shall be encoded into something like "urn:MIME:application%2fmsword."

An alternative could be to define the ContentInfo element as follows:

```
<!ELEMENT ContentInfo (#PCDATA)>
<!ATTLIST ContentInfo
    type        CDATA   #IMPLIED 'urn:MIME'
>

<!-- Examples -->

<ContentInfo>
  application/msword
</ContentInfo>

<ContentInfo type='urn:IOTP'>
  OrderDescription
</ContentInfo>
```

Author Notes:

#99040701 – Allow Resource by value in Manifest

Origin:	John Boyer, Richard D. Brown
Status:	Opened

Comments:

"It may be worth considering the possibility to define a Resource either by means of a locator and a digest or by value."

We may consider the following definition:

```
<!ELEMENT Resource ((Locator, Digest) | Value)>
```

Author Notes:

Adopting this approach will disallow collapsing the Locator element into the Resource element. If the value is provided it does not make a lot of sense to specify a resource location. But, Xlink specification seems to require the href attribute (not clear in the specification).

#99040801 – Add a Certificate Appendix to specification

Origin:	Richard D. Brown
Status:	Opened

Comments:

"It may be worth considering adding a certificate appendix that documents specifics for the different certificate types or certificate locators. As a matter of fact, a LDAP URL might be used but it is not sufficient for locating a particular certificate"

Author Notes:

#99040802 – Add a Security Appendix to specification

Origin:	Richard D. Brown
Status:	Opened

Comments:

"It may be worth considering adding a security appendix such as the one mandated by IETF."

Author Notes:

#99040803 – Add a Compliance Appendix to specification

Origin:	Richard D. Brown
Status:	Opened

Comments:

"It may be worth considering adding an appendix that defines compliance requirements."

Author Notes:

#99040804 – Segregate basic data types

Origin:	Richard D. Brown
Status:	Opened

Comments:

"It may be worth considering segregation of the data types that do not directly relate to XML-DSIG. For example, elements such as Integer, Float, and value might be replaced by making use of some DCD attribute, and others such as IssuerAndSerialNumber or Package might be temporarily moved into some Data DTD. These element definitions will be later superceded by definitions adopted by specialized DTDs.

Author Notes:
