XML Schema Documentation

Table of Contents

- Schema Document Properties
- Global Declarations
  - Element: **CanonicalizationMethod**
  - Element: **DSAKeyValue**
  - Element: **DigestMethod**
  - Element: **DigestValue**
  - Element: **KeyInfo**
  - Element: **KeyName**
  - Element: **KeyValue**
  - Element: **Manifest**
  - Element: **MgmtData**
  - Element: **Object**
  - Element: **PGPData**
  - Element: **RSAKeyValue**
  - Element: **Reference**
  - Element: **RetrievalMethod**
  - Element: **SPKIData**
  - Element: **Signature**
  - Element: **SignatureMethod**
  - Element: **SignatureProperties**
  - Element: **SignatureProperty**
  - Element: **SignatureValue**
  - Element: **SignedInfo**
  - Element: **Transform**
  - Element: **Transforms**
  - Element: **X509Data**
- Global Definitions
  - Complex Type: **CanonicalizationMethodType**
  - Complex Type: **DSAKeyValueType**
  - Complex Type: **DigestMethodType**
  - Complex Type: **KeyInfoType**
  - Complex Type: **KeyValueType**
  - Complex Type: **ManifestType**
  - Complex Type: **ObjectType**
  - Complex Type: **PGPDataType**

- Complex Type: **RSAKeyValueType**
- Complex Type: **ReferenceType**
- Complex Type: **RetrievalMethodType**
- Complex Type: **SPKIDataType**
- Complex Type: **SignatureMethodType**
- Complex Type: **SignaturePropertiesType**
- Complex Type: **SignaturePropertyType**
- Complex Type: **SignatureType**
- Complex Type: **SignatureValueType**
- Complex Type: **SignedInfoType**
- Complex Type: **TransformType**
- Complex Type: **TransformsType**
- Complex Type: **X509DataType**
- Complex Type: **X509IssuerSerialType**
- Simple Type: **CryptoBinary**
- Simple Type: **DigestValueType**
- Simple Type: **HMACOutputLengthType**

- Legend
- Glossary

top

Schema Document Properties

Target Namespace	**http://www.w3.org/2000/09/xmldsig#**
Version	0.1
Element and Attribute Namespaces	• Global element and attribute declarations belong to this schema's target namespace. • By default, local element declarations belong to this schema's target namespace. • By default, local attribute declarations have no namespace.

Declared Namespaces

Prefix	Namespace
Default namespace	http://www.w3.org/2001/XMLSchema
xml	http://www.w3.org/XML/1998/namespace
ds	**http://www.w3.org/2000/09/xmldsig#**

Schema Component Representation

```
<schema targetNamespace="http://www.w3.org/2000/09/xmldsig#"
elementFormDefault="qualified" version="0.1">
```

```
   ...
</schema>
```

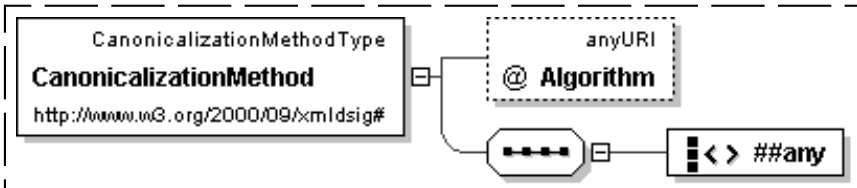
top

Global Declarations

Element: CanonicalizationMethod

Name	CanonicalizationMethod
Type	ds:CanonicalizationMethodType
Nillable	no
Abstract	no

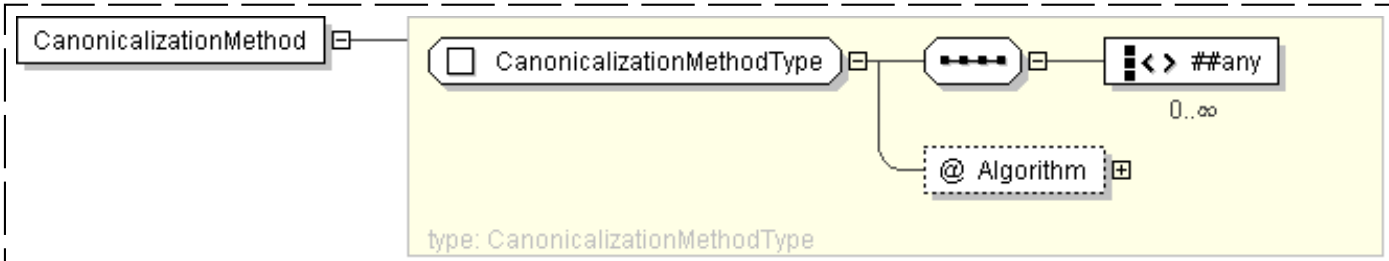
Logical Diagram



XML Instance Representation

```
<ds:CanonicalizationMethod
Algorithm="anyURI [1]">
<!-- Mixed content -->
   Allow any elements from any namespace (strict validation). [0..*]
</ds:CanonicalizationMethod>
```

Diagram



Schema Component Representation

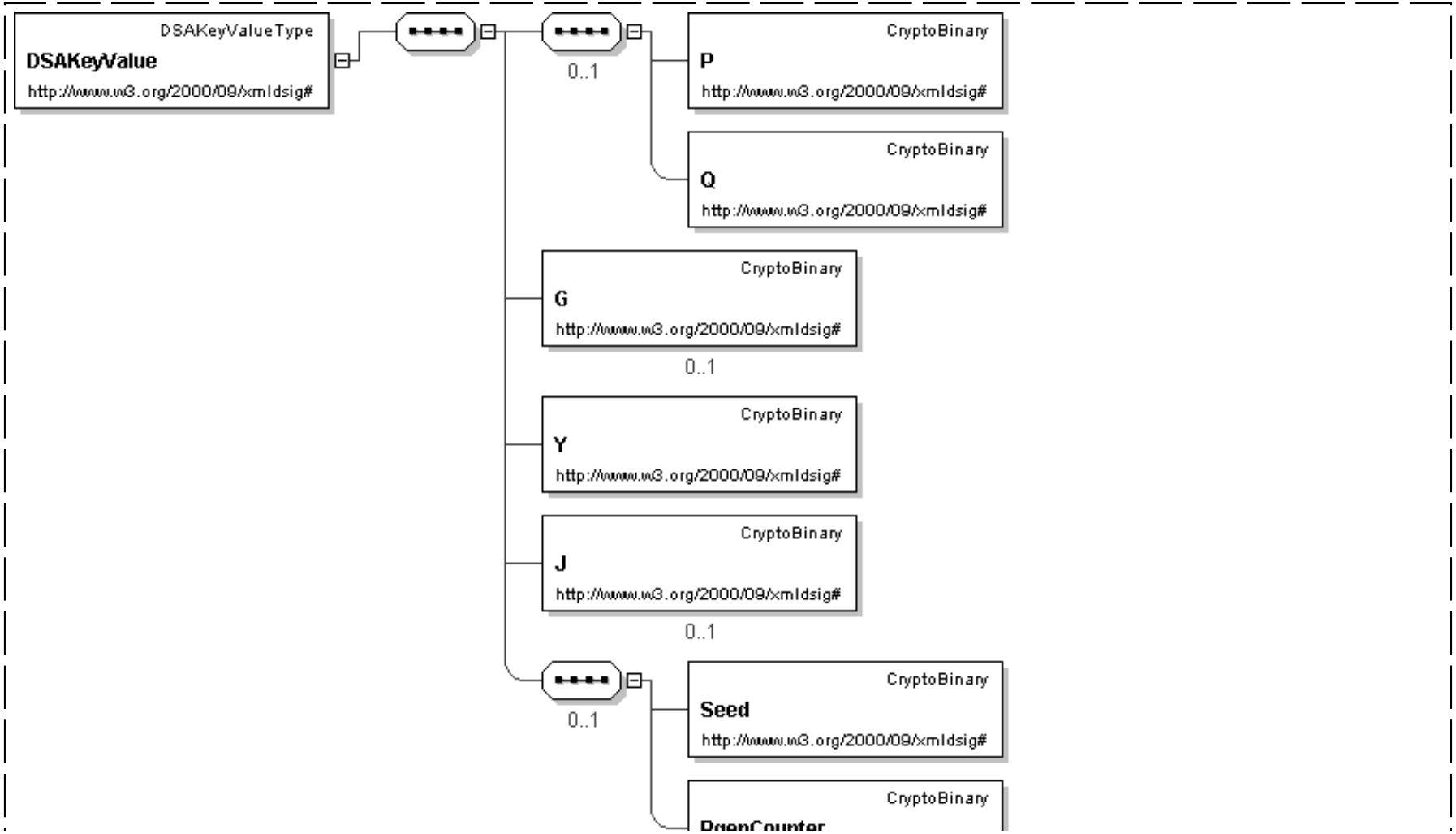
```
<element name="CanonicalizationMethod" type=" ds:CanonicalizationMethodType "/>
```

top

Element: DSAKeyValue

Name	DSAKeyValue
Type	ds:DSAKeyValueType
Nillable	no
Abstract	no

Logical Diagram

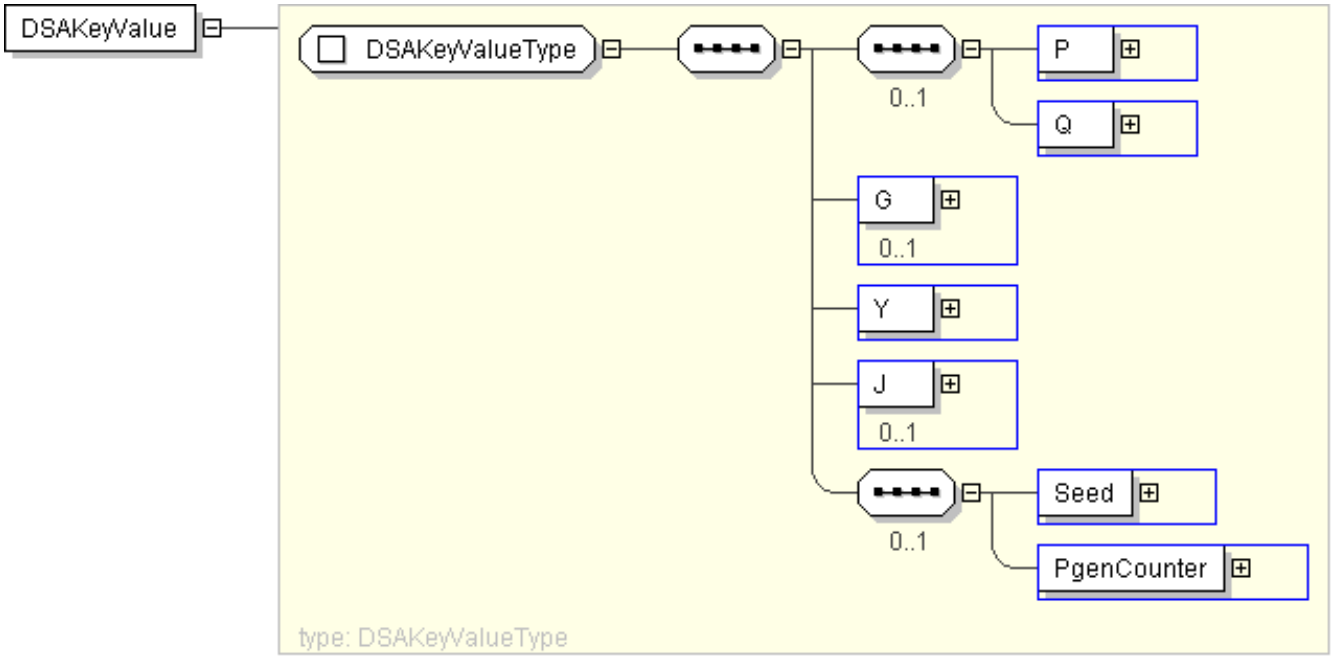


http://www.w3.org/2000/09/xmldsig#

XML Instance Representation

```
<ds:DSAKeyValue>
Start Sequence [0..1]
      <ds:P> ds:CryptoBinary </ds:P> [1]
      <ds:Q> ds:CryptoBinary </ds:Q> [1]
End Sequence
   <ds:G> ds:CryptoBinary </ds:G> [0..1]
   <ds:Y> ds:CryptoBinary </ds:Y> [1]
   <ds:J> ds:CryptoBinary </ds:J> [0..1]
Start Sequence [0..1]
      <ds:Seed> ds:CryptoBinary </ds:Seed> [1]
      <ds:PgenCounter> ds:CryptoBinary </ds:PgenCounter> [1]
End Sequence
</ds:DSAKeyValue>
```

Diagram



Schema Component Representation

```
<element name="DSAKeyValue" type=" ds:DSAKeyValueType "/>
```

top

Element: DigestMethod

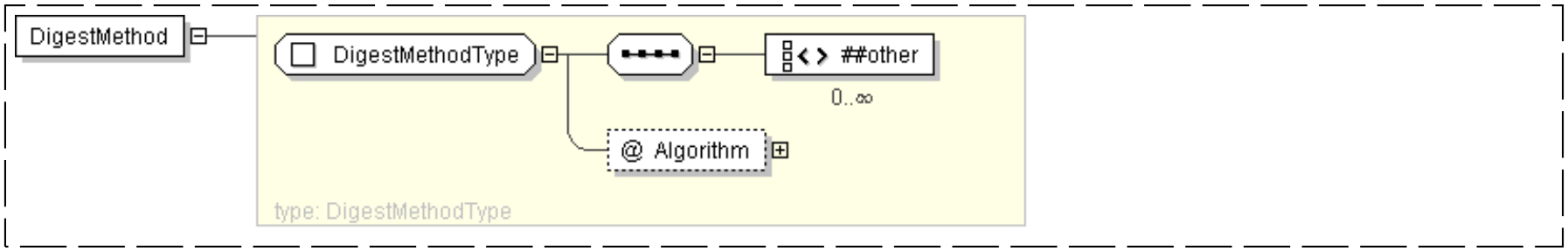
Name	DigestMethod
Type	ds:DigestMethodType
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation

```
<ds:DigestMethod
Algorithm="anyURI [1]">
<!-- Mixed content -->
   Allow any elements from a namespace other than this schema's namespace (lax validation). [0..*]
</ds:DigestMethod>
```

Diagram



Schema Component Representation

```
<element name="DigestMethod" type=" ds:DigestMethodType "/>
```

top

Element: DigestValue

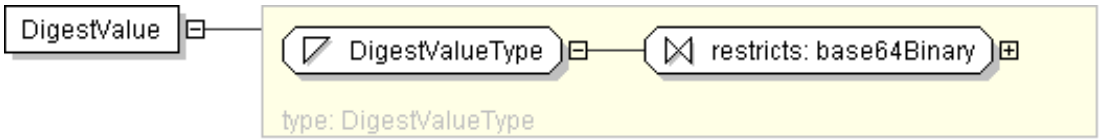
Name	DigestValue
Type	ds:DigestValueType
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation

```
<ds:DigestValue> ds:DigestValueType </ds:DigestValue>
```

Diagram



Schema Component Representation

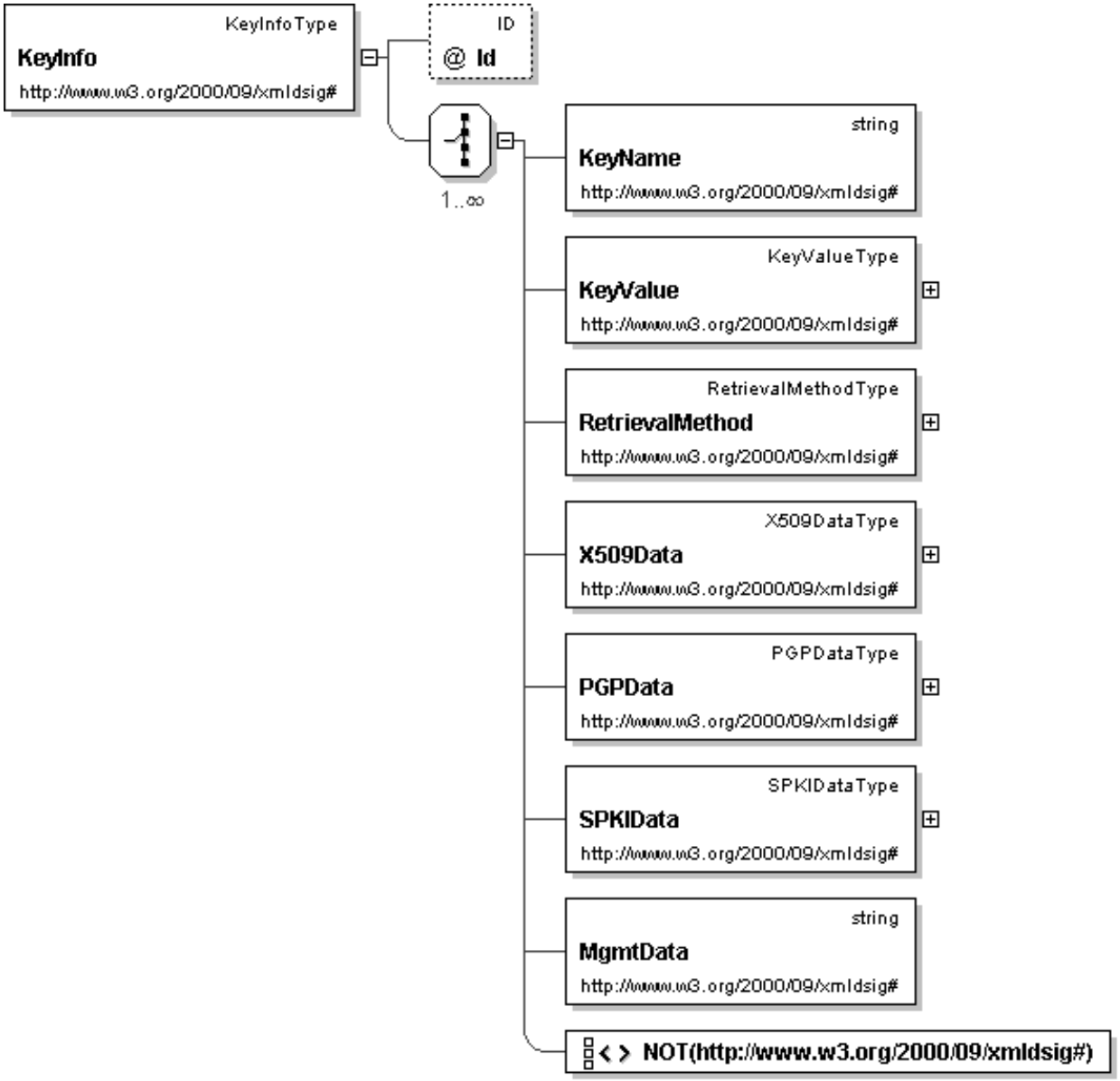
```
<element name="DigestValue" type=" ds:DigestValueType "/>
```

top

Element: KeyInfo

Name	KeyInfo
Type	ds:KeyInfoType
Nillable	no
Abstract	no

Logical Diagram

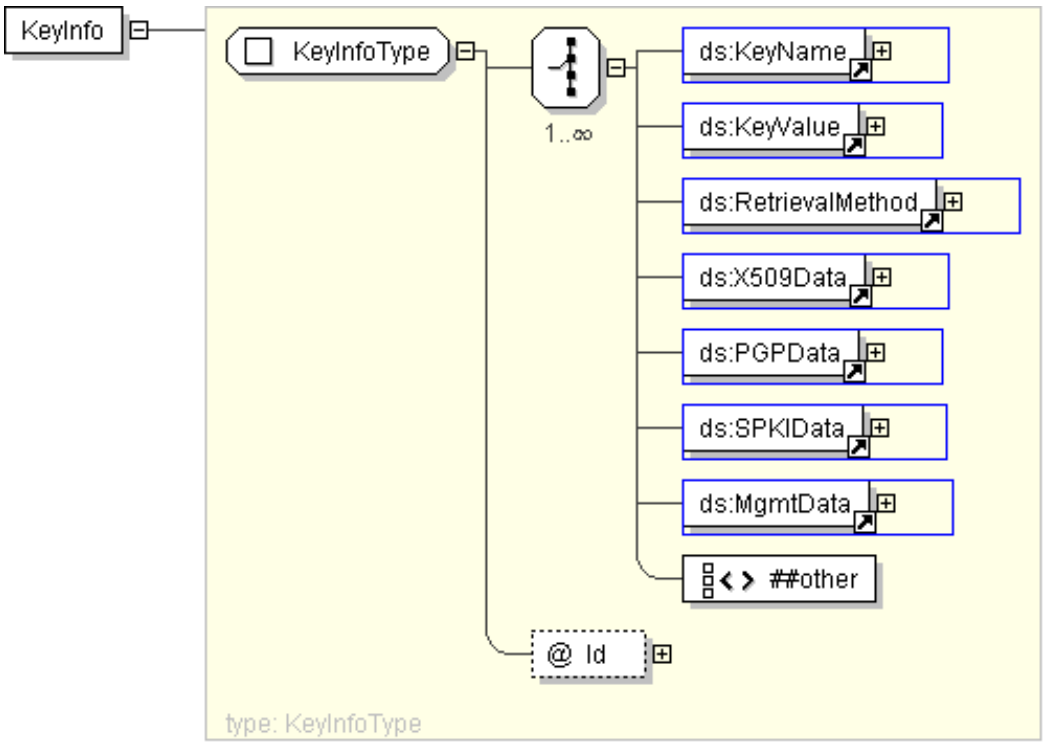


XML Instance Representation

```
<ds:KeyInfo
Id="ID [0..1]">
<!-- Mixed content -->
Start Choice [1..*]
      <ds:KeyName> ... </ds:KeyName> [1]
      <ds:KeyValue> ... </ds:KeyValue> [1]
```

```
        <ds:RetrievalMethod> ... </ds:RetrievalMethod> [1]
        <ds:X509Data> ... </ds:X509Data> [1]
        <ds:PGPData> ... </ds:PGPData> [1]
        <ds:SPKIData> ... </ds:SPKIData> [1]
        <ds:MgmtData> ... </ds:MgmtData> [1]
        Allow any elements from a namespace other than this schema's namespace (lax validation). [1]
End Choice
</ds:KeyInfo>
```

Diagram



Schema Component Representation

```
<element name="KeyInfo" type=" ds:KeyInfoType "/>
```

top

Element: KeyName

Name	KeyName

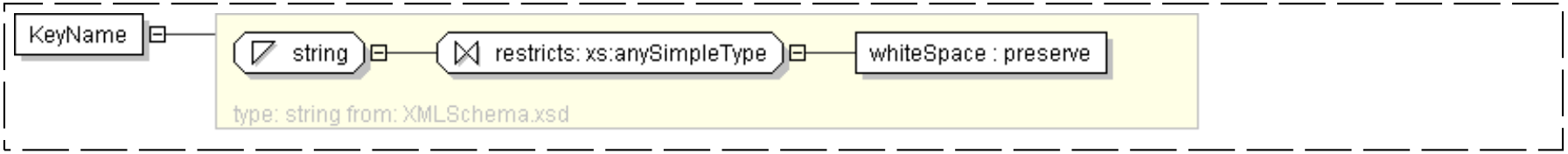
Type	string
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation

```
<ds:KeyName> string </ds:KeyName>
```

Diagram



Schema Component Representation

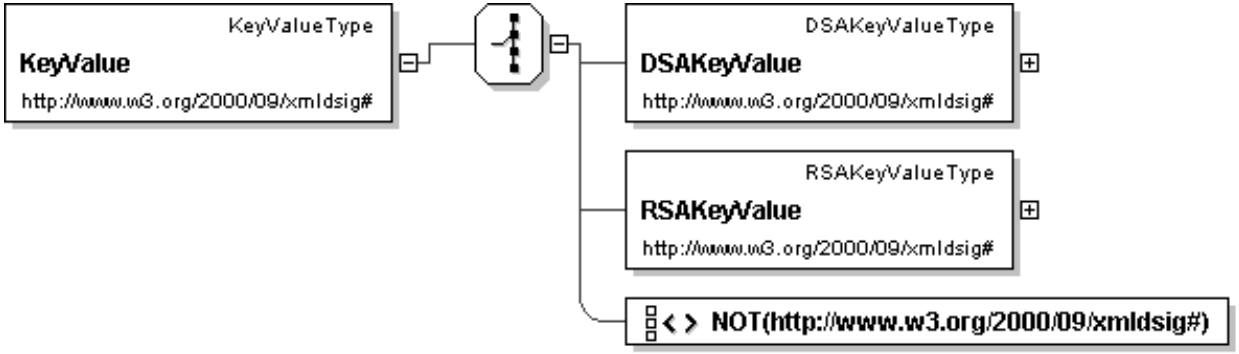
```
<element name="KeyName" type=" string "/>
```

top

Element: KeyValue

Name	KeyValue
Type	ds:KeyValueType
Nillable	no
Abstract	no

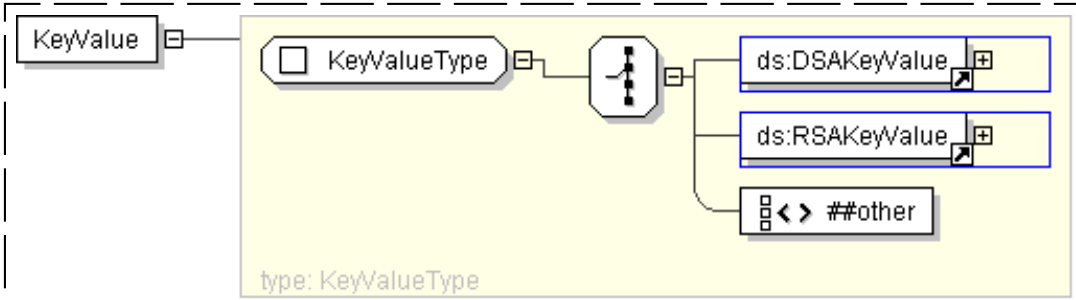
Logical Diagram



XML Instance Representation

```
<ds:KeyValue>
<!-- Mixed content -->
Start Choice [1]
      <ds:DSAKeyValue> ... </ds:DSAKeyValue> [1]
      <ds:RSAKeyValue> ... </ds:RSAKeyValue> [1]
      Allow any elements from a namespace other than this schema's namespace (lax validation). [1]
End Choice
</ds:KeyValue>
```

Diagram



Schema Component Representation

```
<element name="KeyValue" type=" ds:KeyValueType "/>
```

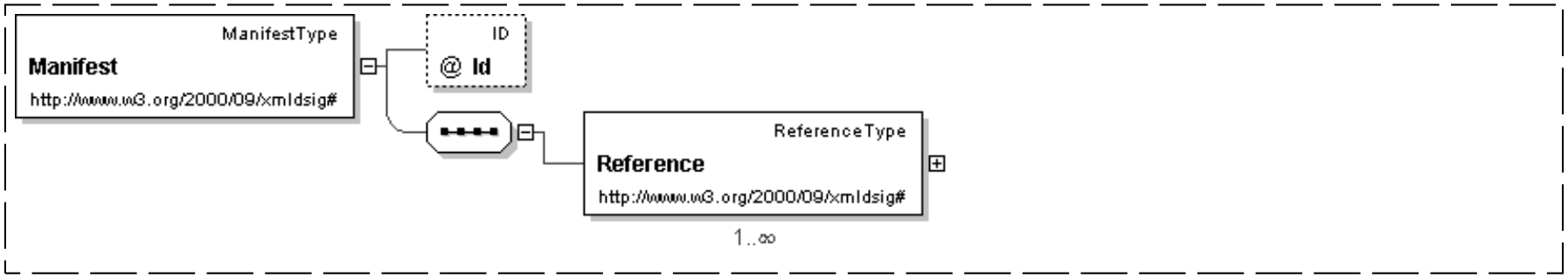
top

Element: Manifest

Name	Manifest

Type	ds:ManifestType
Nillable	no
Abstract	no

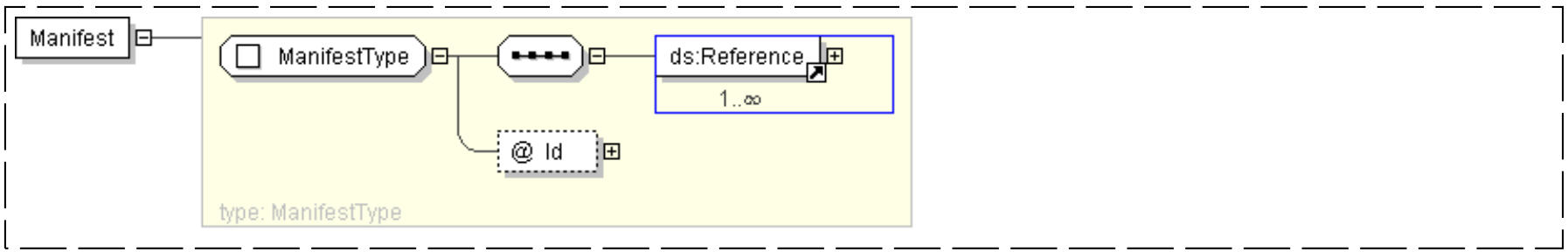
Logical Diagram



XML Instance Representation

```
<ds:Manifest
Id="ID [0..1]">
   <ds:Reference> ... </ds:Reference> [1..*]
</ds:Manifest>
```

Diagram



Schema Component Representation

```
<element name="Manifest" type=" ds:ManifestType "/>
```

top

Element: MgmtData

Name	MgmtData
Type	string

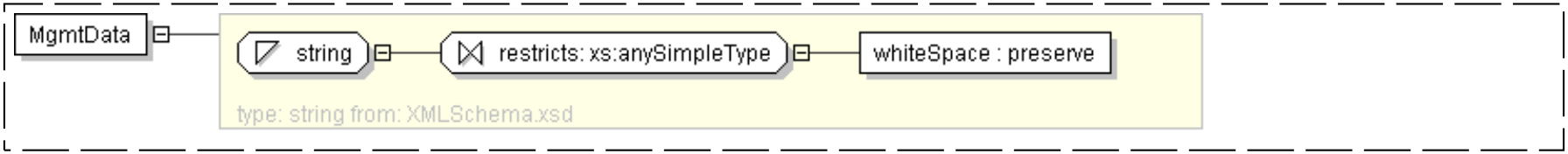
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation

```
<ds:MgmtData> string </ds:MgmtData>
```

Diagram



Schema Component Representation

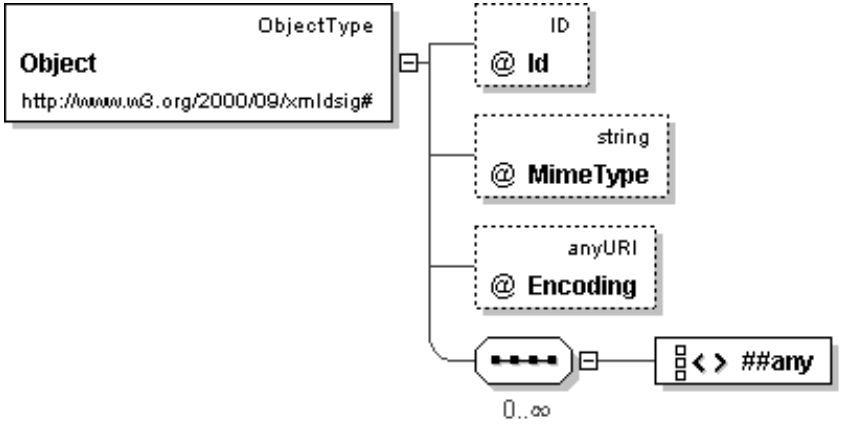
```
<element name="MgmtData" type=" string "/>
```

top

Element: Object

Name	Object
Type	ds:ObjectType
Nillable	no
Abstract	no

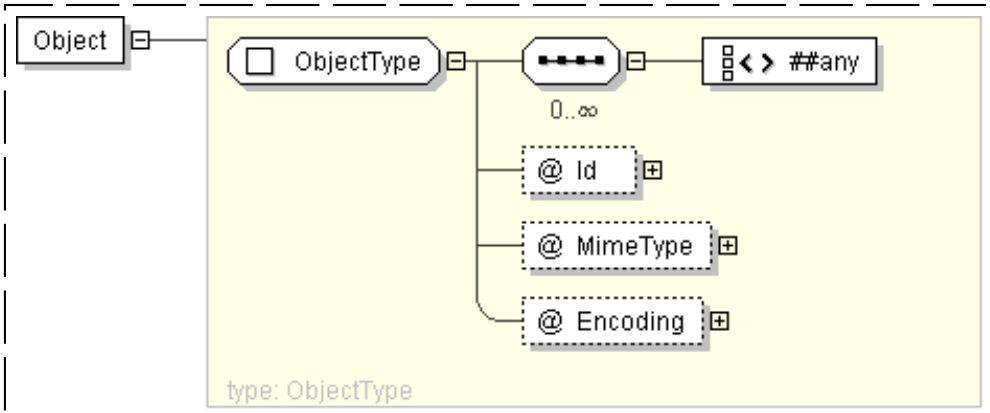
Logical Diagram



XML Instance Representation

```
<ds:Object
Id="ID [0..1]"
MimeType="string [0..1]"
Encoding="anyURI [0..1]">
<!-- Mixed content -->
Start Sequence [0..*]
      Allow any elements from any namespace (lax validation). [1]
End Sequence
</ds:Object>
```

Diagram



Schema Component Representation

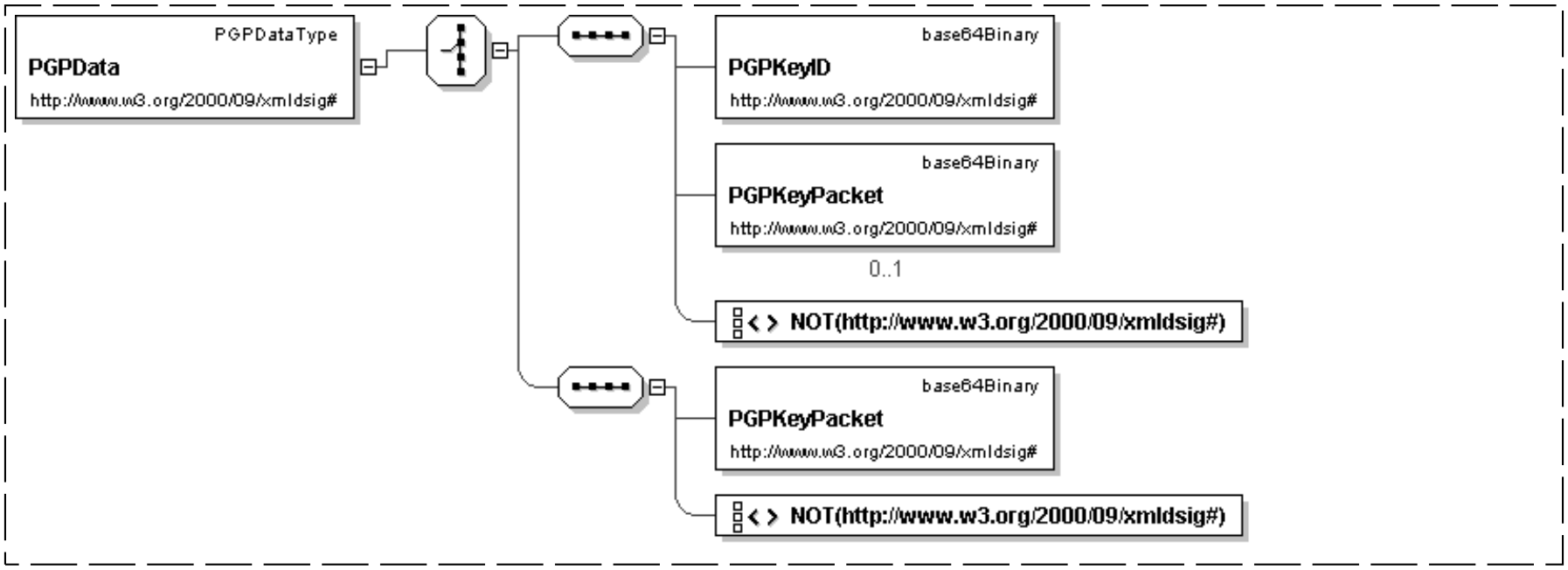
```
<element name="Object" type=" ds:ObjectType "/>
```

top

Element: PGPData

Name	PGPData
Type	ds:PGPDataType
Nillable	no
Abstract	no

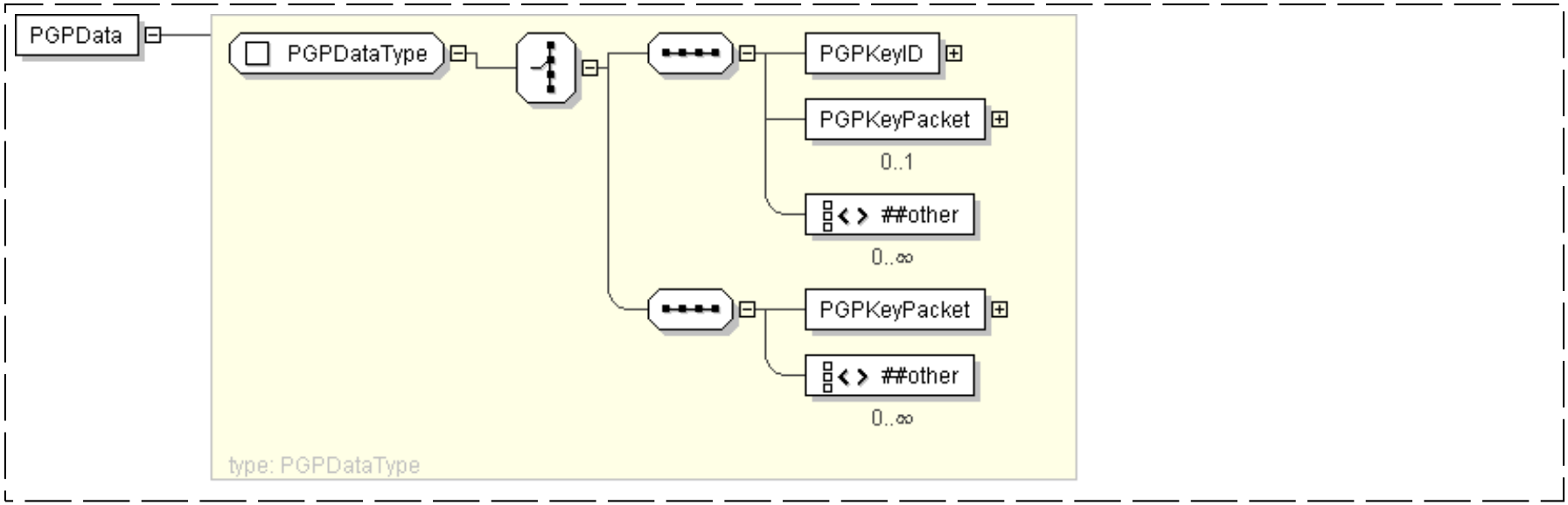
Logical Diagram



XML Instance Representation

```
<ds:PGPData>
Start Choice [1]
      <ds:PGPKeyID> base64Binary </ds:PGPKeyID> [1]
      <ds:PGPKeyPacket> base64Binary </ds:PGPKeyPacket> [0..1]
      Allow any elements from a namespace other than this schema's namespace (lax validation). [0..*]
      <ds:PGPKeyPacket> base64Binary </ds:PGPKeyPacket> [1]
      Allow any elements from a namespace other than this schema's namespace (lax validation). [0..*]
End Choice
</ds:PGPData>
```

Diagram



Schema Component Representation

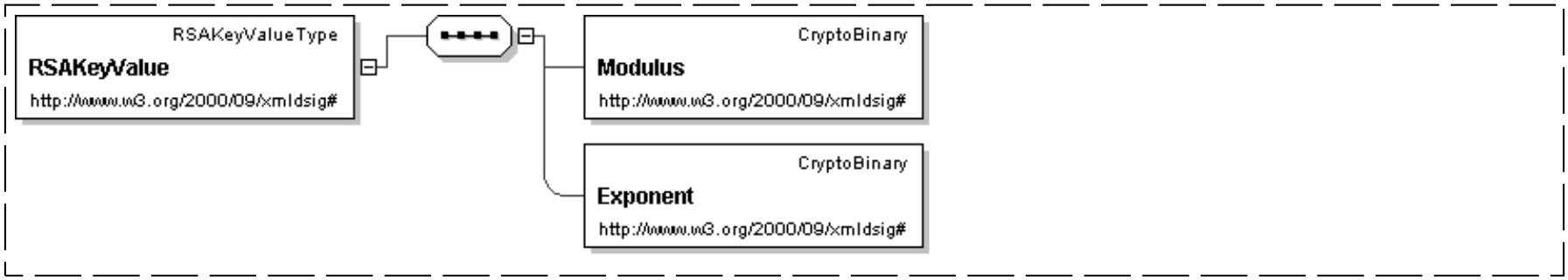
```
<element name="PGPData" type=" ds:PGPDataType "/>
```

top

Element: RSAKeyValue

Name	RSAKeyValue
Type	ds:RSAKeyValueType
Nillable	no
Abstract	no

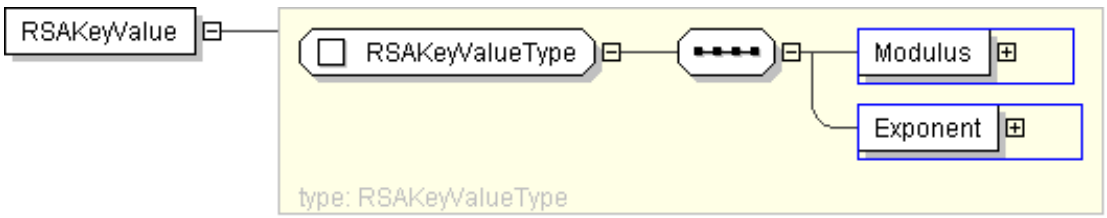
Logical Diagram



XML Instance Representation

```
<ds:RSAKeyValue>
   <ds:Modulus> ds:CryptoBinary </ds:Modulus> [1]
   <ds:Exponent> ds:CryptoBinary </ds:Exponent> [1]
</ds:RSAKeyValue>
```

Diagram



Schema Component Representation

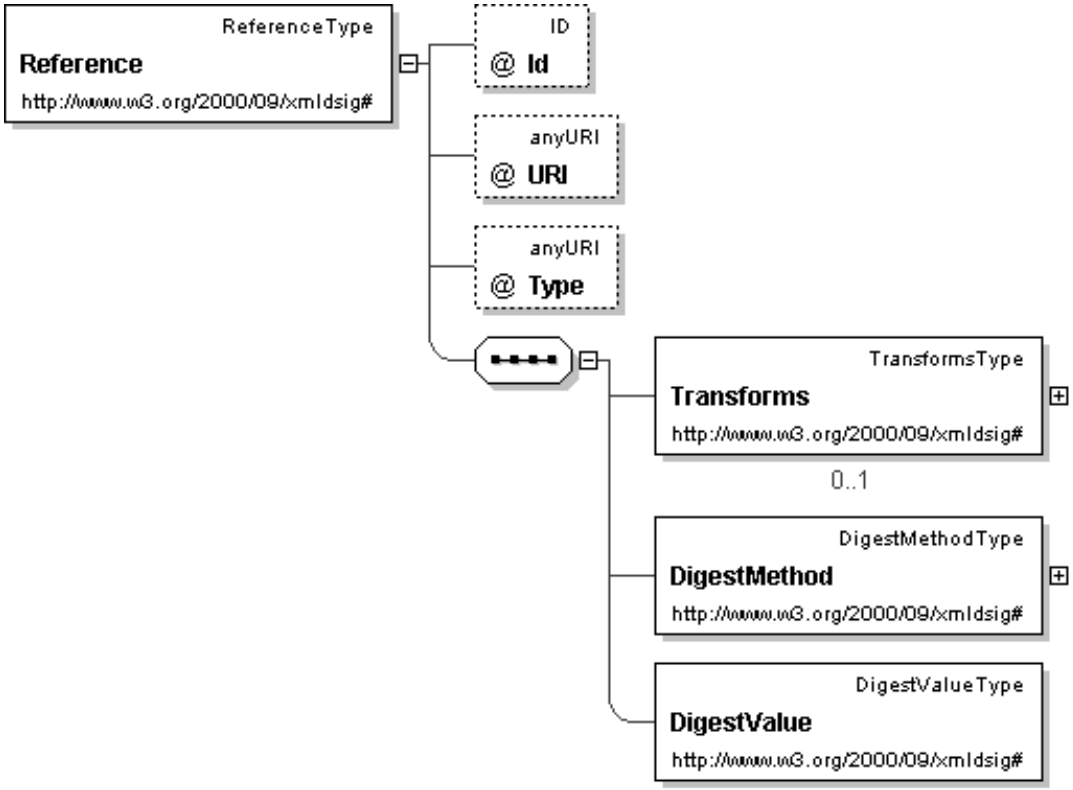
```
<element name="RSAKeyValue" type=" ds:RSAKeyValueType "/>
```

top

Element: Reference

Name	Reference
Type	ds:ReferenceType
Nillable	no
Abstract	no

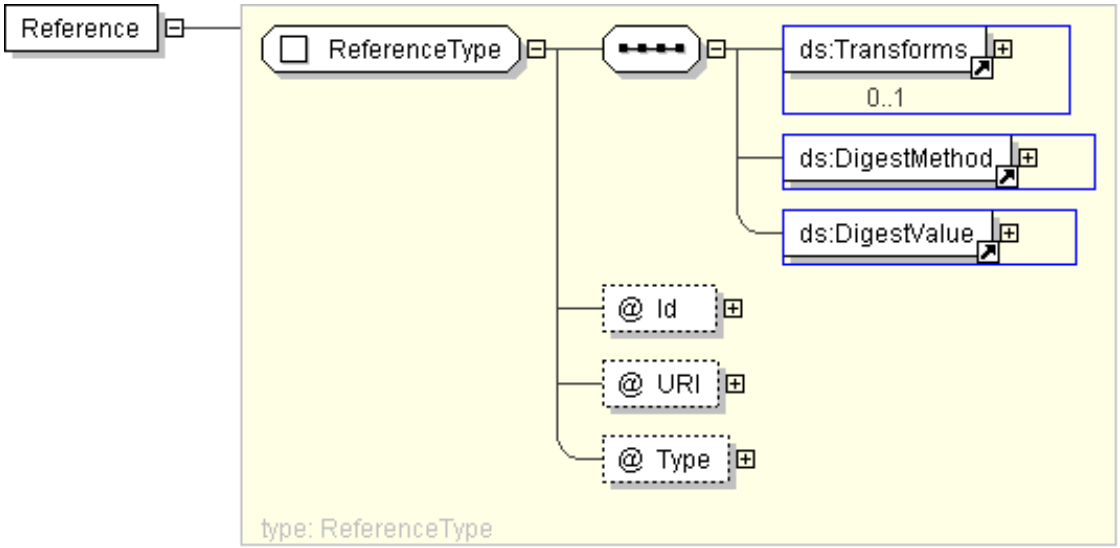
Logical Diagram



XML Instance Representation

```
<ds:Reference
Id="ID [0..1]"
URI="anyURI [0..1]"
Type="anyURI [0..1]">
   <ds:Transforms> ... </ds:Transforms> [0..1]
   <ds:DigestMethod> ... </ds:DigestMethod> [1]
   <ds:DigestValue> ... </ds:DigestValue> [1]
</ds:Reference>
```

Diagram



Schema Component Representation

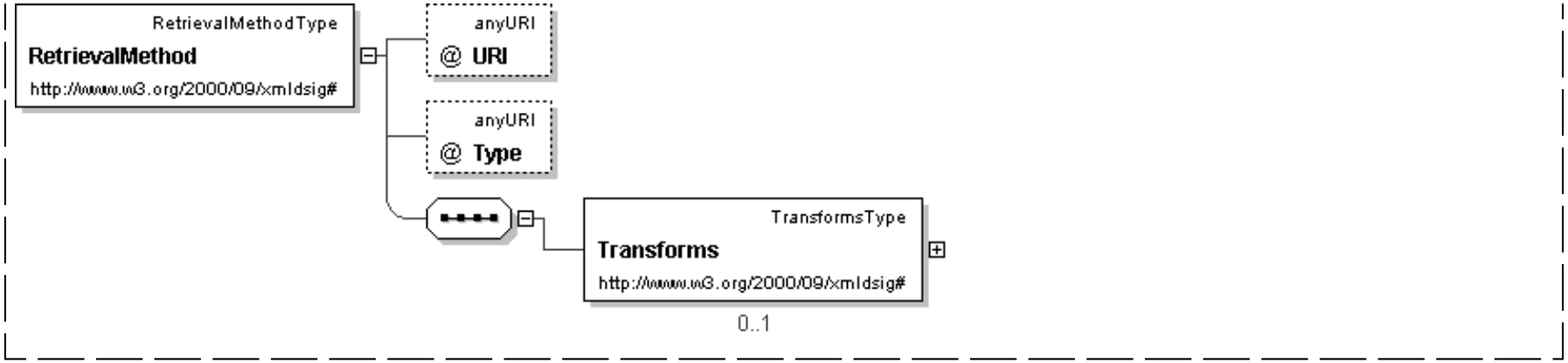
```
<element name="Reference" type=" ds:ReferenceType "/>
```

top

Element: RetrievalMethod

Name	RetrievalMethod
Type	ds:RetrievalMethodType
Nillable	no
Abstract	no

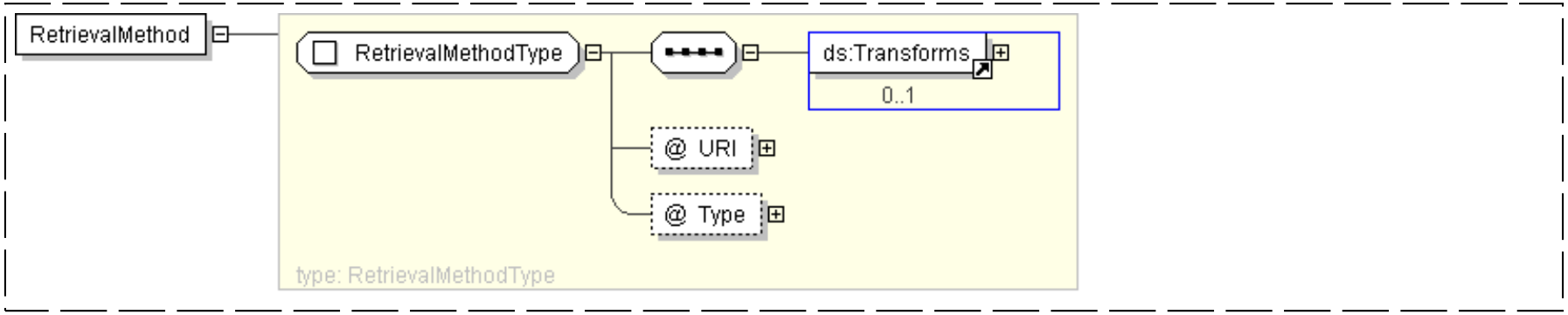
Logical Diagram



XML Instance Representation

```
<ds:RetrievalMethod
URI="anyURI [0..1]"
Type="anyURI [0..1]">
   <ds:Transforms> ... </ds:Transforms> [0..1]
</ds:RetrievalMethod>
```

Diagram



Schema Component Representation

```
<element name="RetrievalMethod" type=" ds:RetrievalMethodType "/>
```

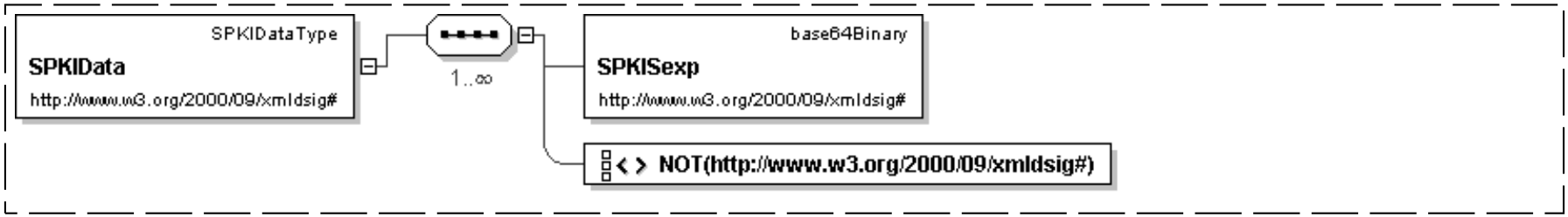
top

Element: SPKIData

Name	SPKIData

Type	ds:SPKIDataType
Nillable	no
Abstract	no

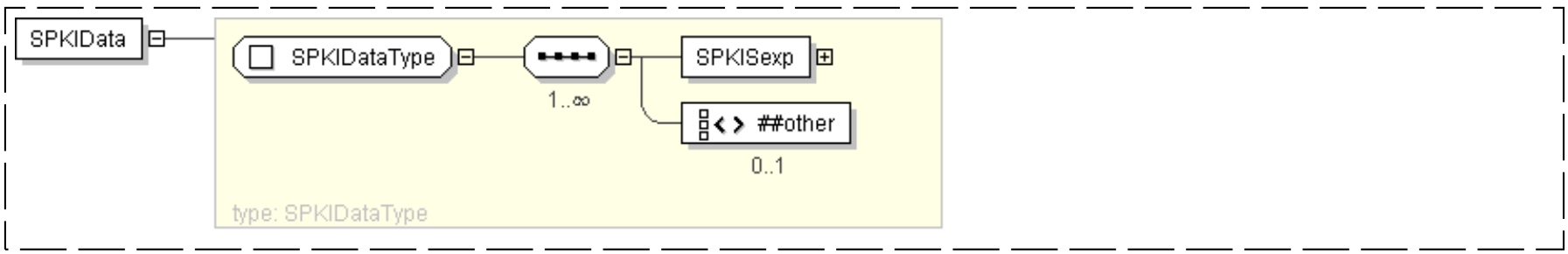
Logical Diagram



XML Instance Representation

```
<ds:SPKIData>
Start Sequence [1..*]
      <ds:SPKISexp> base64Binary </ds:SPKISexp> [1]
      Allow any elements from a namespace other than this schema's namespace (lax validation). [0..1]
End Sequence
</ds:SPKIData>
```

Diagram



Schema Component Representation

```
<element name="SPKIData" type=" ds:SPKIDataType "/>
```

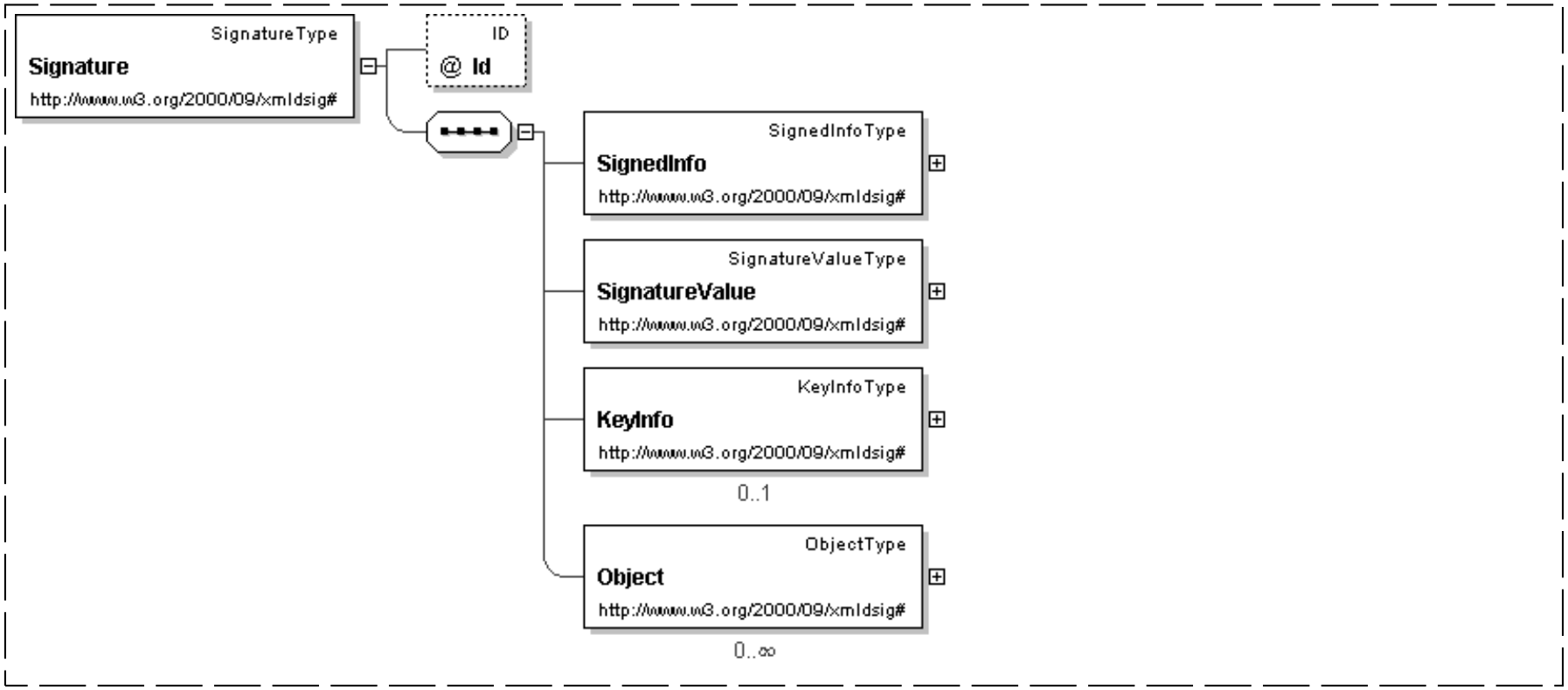
top

Element: Signature

Name	Signature

Type	ds:SignatureType
Nillable	no
Abstract	no

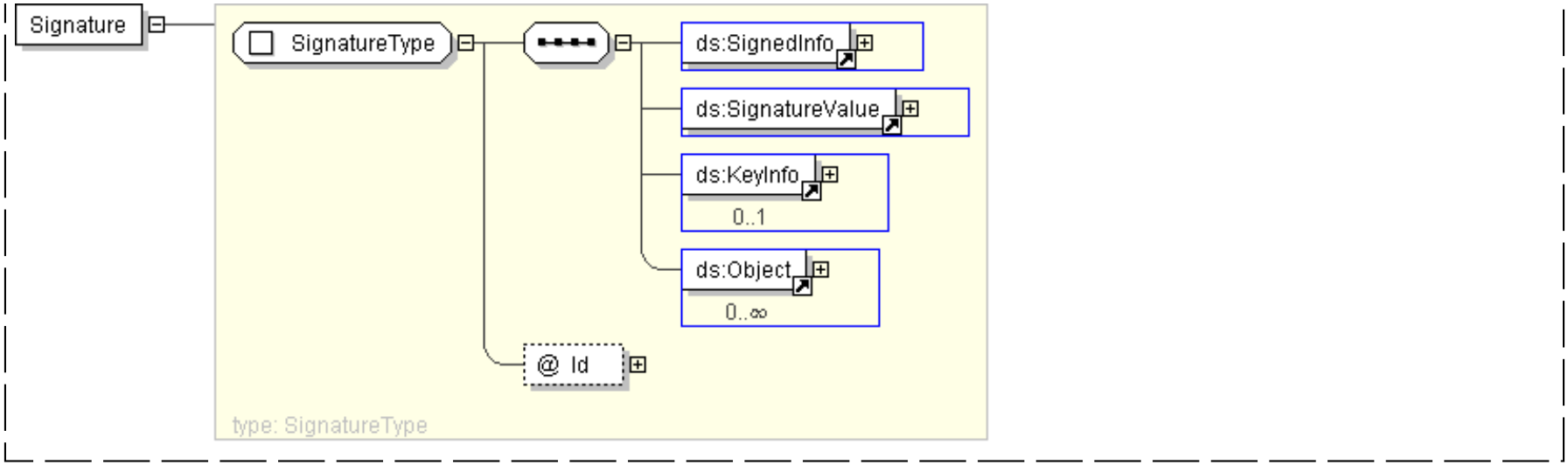
Logical Diagram



XML Instance Representation

```
<ds:Signature
Id="ID [0..1]">
   <ds:SignedInfo> ... </ds:SignedInfo> [1]
   <ds:SignatureValue> ... </ds:SignatureValue> [1]
   <ds:KeyInfo> ... </ds:KeyInfo> [0..1]
   <ds:Object> ... </ds:Object> [0..*]
</ds:Signature>
```

Diagram



Schema Component Representation

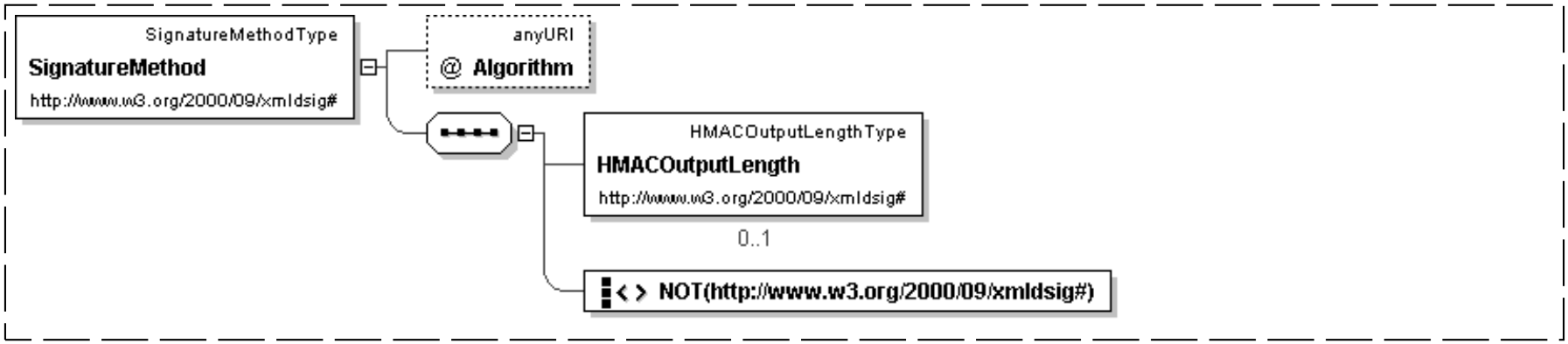
```
<element name="Signature" type=" ds:SignatureType "/>
```

top

Element: SignatureMethod

Name	SignatureMethod
Type	ds:SignatureMethodType
Nillable	no
Abstract	no

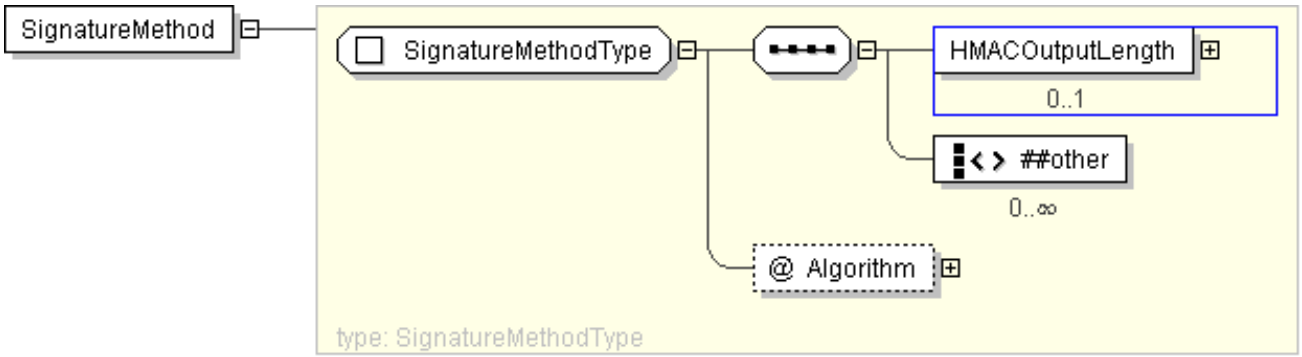
Logical Diagram



XML Instance Representation

```
<ds:SignatureMethod
Algorithm="anyURI [1]">
<!-- Mixed content -->
   <ds:HMACOutputLength> ds:HMACOutputLengthType </ds:HMACOutputLength> [0..1]
   Allow any elements from a namespace other than this schema's namespace (strict validation).
   [0..*]
</ds:SignatureMethod>
```

Diagram



Schema Component Representation

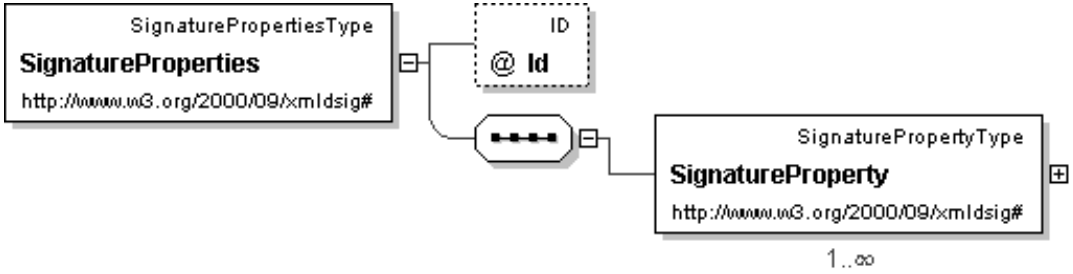
```
<element name="SignatureMethod" type=" ds:SignatureMethodType "/>
```

top

Element: SignatureProperties

Name	SignatureProperties
Type	ds:SignaturePropertiesType
Nillable	no
Abstract	no

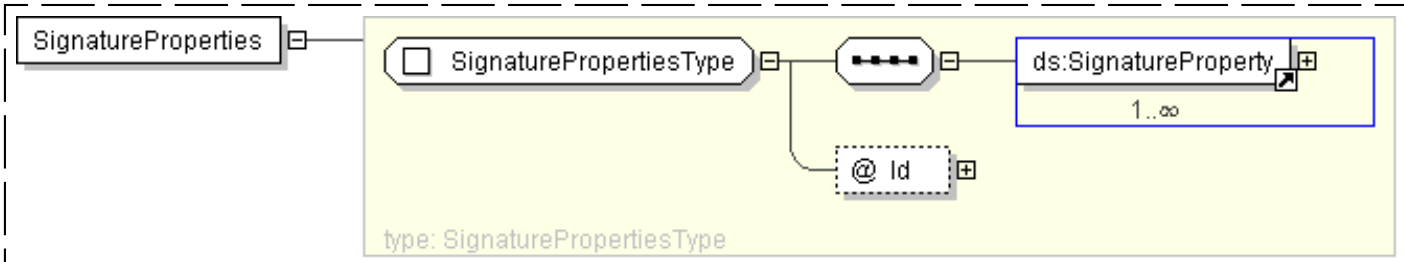
Logical Diagram



XML Instance Representation

```
<ds:SignatureProperties
Id="ID [0..1]">
   <ds:SignatureProperty> ... </ds:SignatureProperty> [1..*]
</ds:SignatureProperties>
```

Diagram



Schema Component Representation

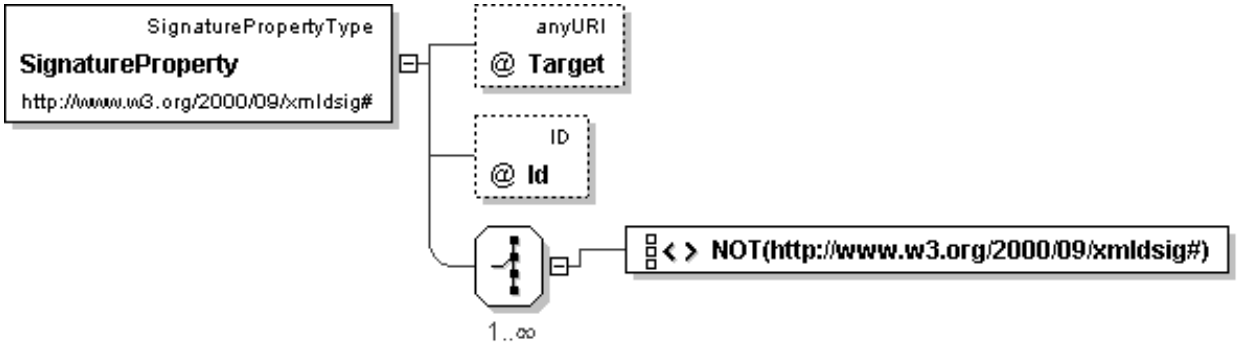
```
<element name="SignatureProperties" type=" ds:SignaturePropertiesType "/>
```

top

Element: SignatureProperty

Name	SignatureProperty
Type	ds:SignaturePropertyType
Nillable	no
Abstract	no

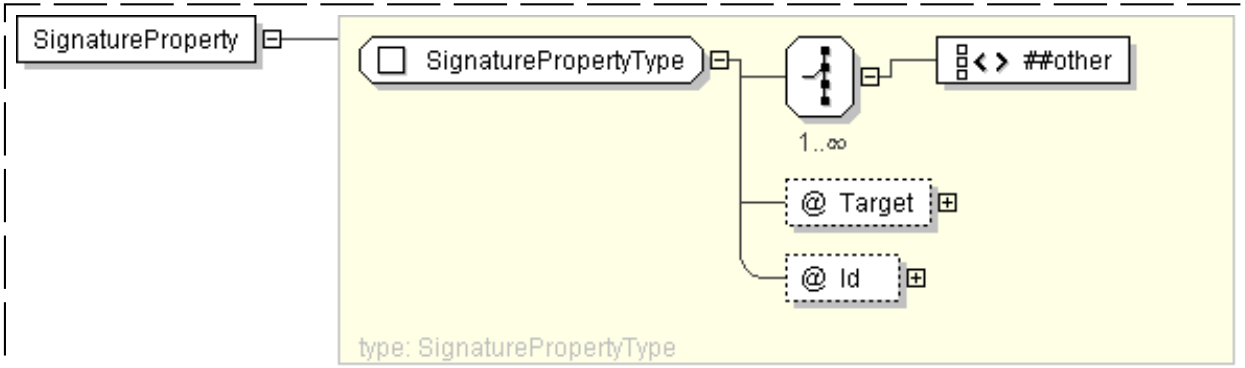
Logical Diagram



XML Instance Representation

```
<ds:SignatureProperty
Target="anyURI [1]"
Id="ID [0..1]">
<!-- Mixed content -->
Start Choice [1..*]
      Allow any elements from a namespace other than this schema's namespace (lax validation). [1]
End Choice
</ds:SignatureProperty>
```

Diagram



Schema Component Representation

```
<element name="SignatureProperty" type=" ds:SignaturePropertyType "/>
```

top

Element: SignatureValue

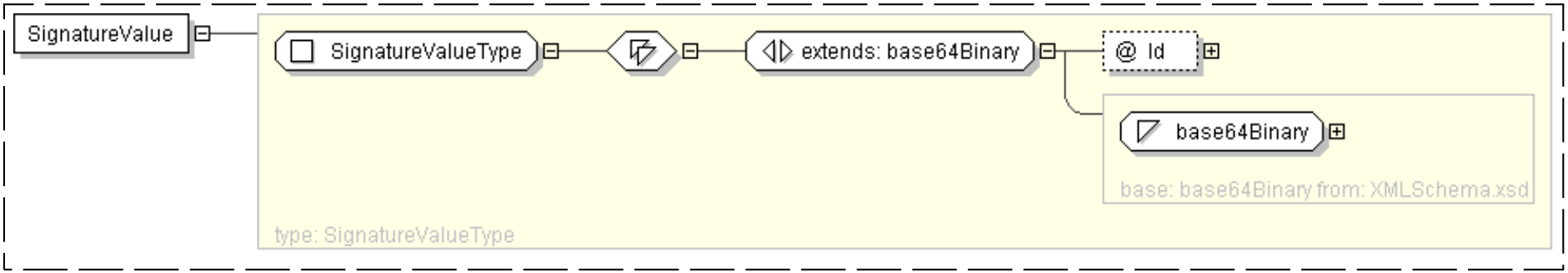
Name	SignatureValue
Type	ds:SignatureValueType
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation

```
<ds:SignatureValue
Id="ID [0..1]">
base64Binary
</ds:SignatureValue>
```

Diagram



Schema Component Representation

```
<element name="SignatureValue" type=" ds:SignatureValueType "/>
```

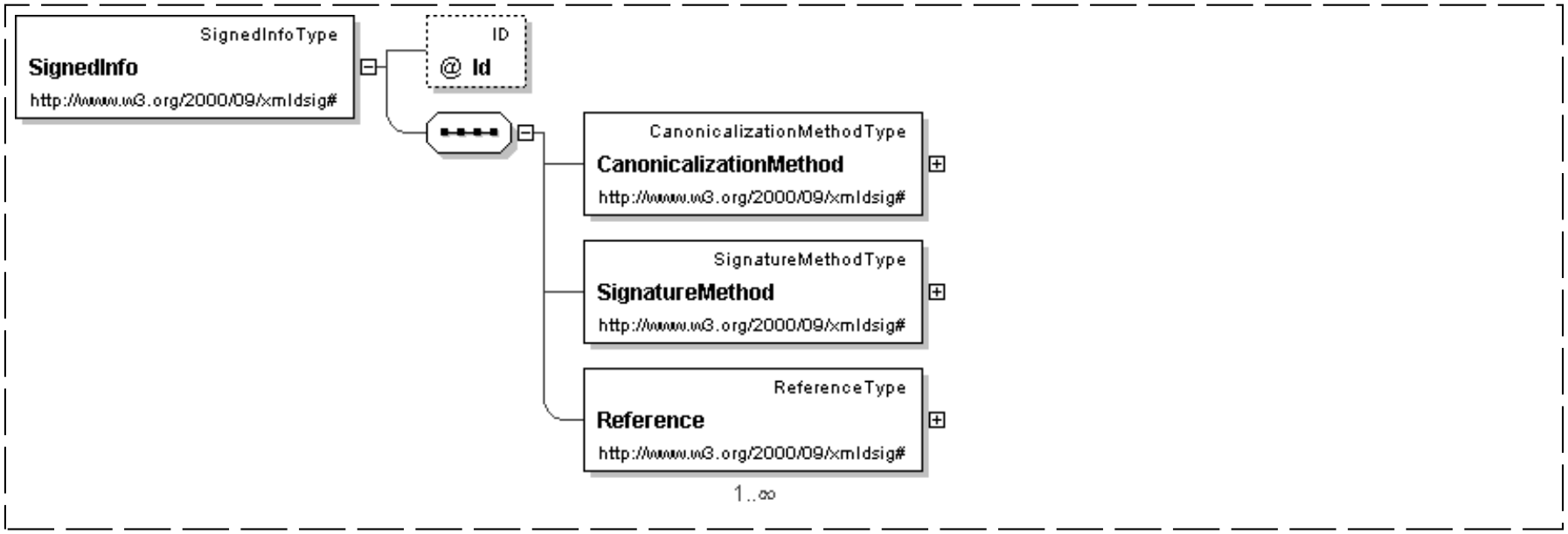
top

Element: SignedInfo

Name	SignedInfo
Type	ds:SignedInfoType
Nillable	no

Abstract	no

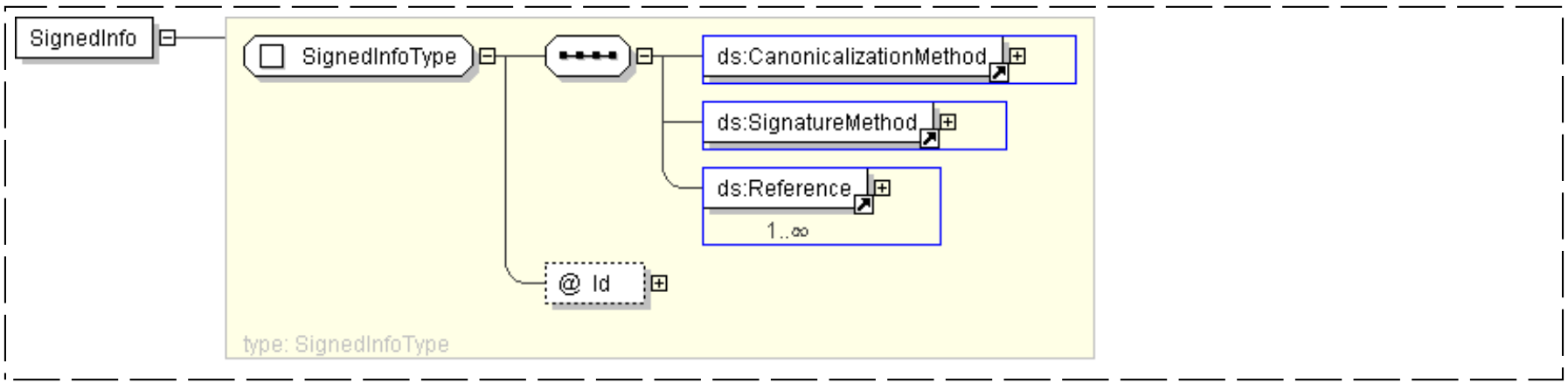
Logical Diagram



XML Instance Representation

```
<ds:SignedInfo
Id="ID [0..1]">
   <ds:CanonicalizationMethod> ... </ds:CanonicalizationMethod> [1]
   <ds:SignatureMethod> ... </ds:SignatureMethod> [1]
   <ds:Reference> ... </ds:Reference> [1..*]
</ds:SignedInfo>
```

Diagram



Schema Component Representation

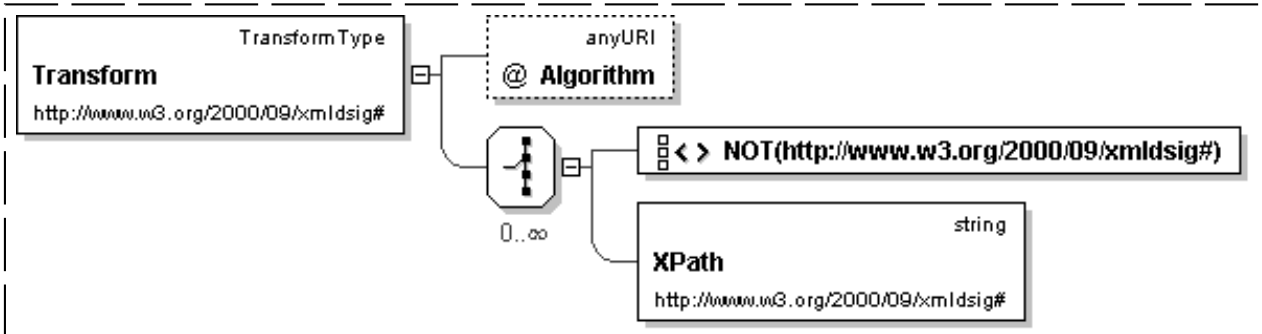
```
<element name="SignedInfo" type=" ds:SignedInfoType "/>
```

top

Element: Transform

Name	Transform
Type	ds:TransformType
Nillable	no
Abstract	no

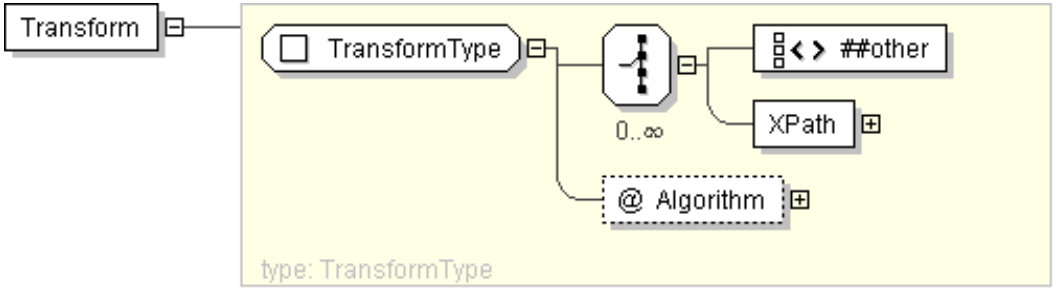
Logical Diagram



XML Instance Representation

```
<ds:Transform
Algorithm="anyURI [1]">
<!-- Mixed content -->
Start Choice [0..*]
      Allow any elements from a namespace other than this schema's namespace (lax validation). [1]
      <ds:XPath> string </ds:XPath> [1]
End Choice
</ds:Transform>
```

Diagram



Schema Component Representation

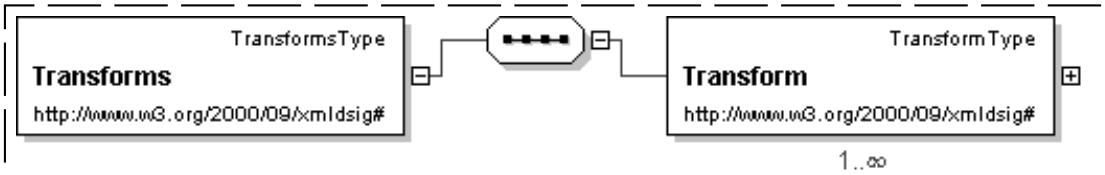
```
<element name="Transform" type=" ds:TransformType "/>
```

top

Element: Transforms

Name	Transforms
Type	ds:TransformsType
Nillable	no
Abstract	no

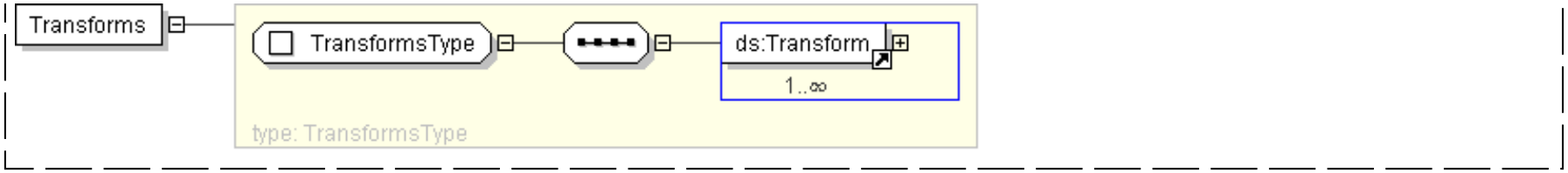
Logical Diagram



XML Instance Representation

```
<ds:Transforms>
   <ds:Transform> ... </ds:Transform> [1..*]
</ds:Transforms>
```

Diagram



Schema Component Representation

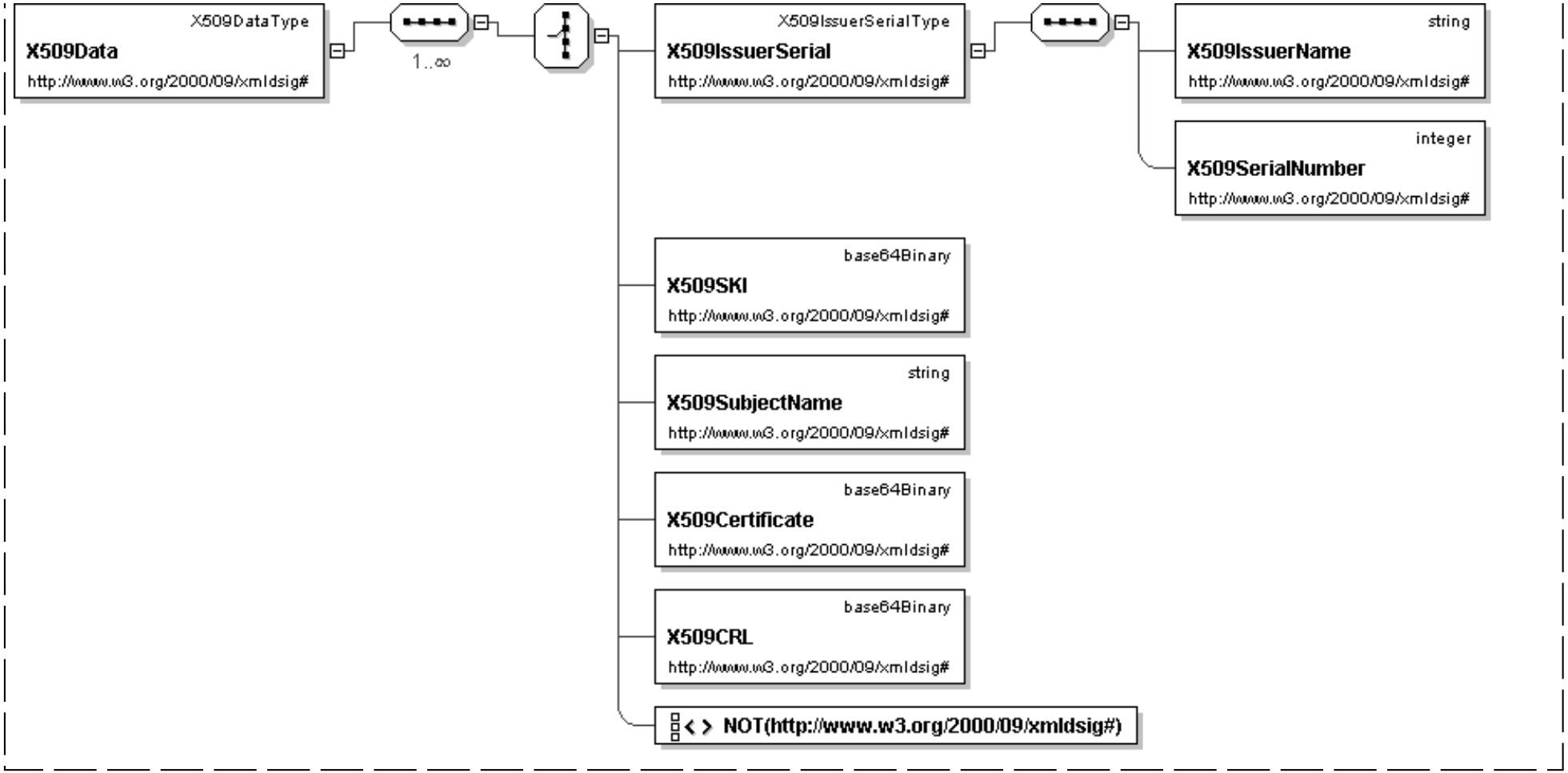
```
<element name="Transforms" type=" ds:TransformsType "/>
```

top

Element: X509Data

Name	X509Data
Type	ds:X509DataType
Nillable	no
Abstract	no

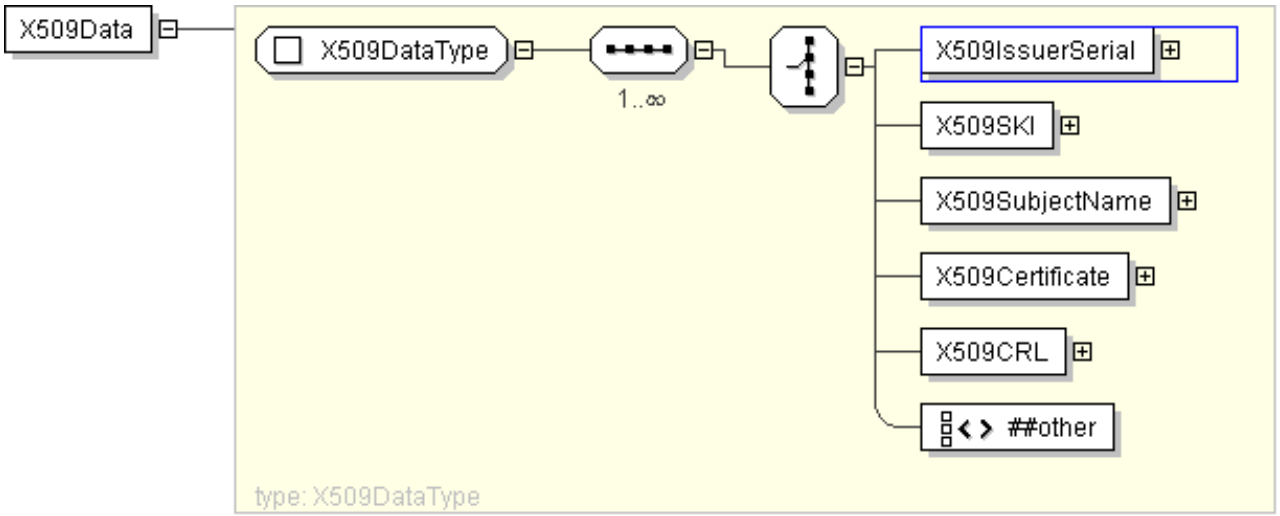
Logical Diagram



XML Instance Representation

```
<ds:X509Data>
Start Sequence [1..*]
Start Choice [1]
        <ds:X509IssuerSerial> ds:X509IssuerSerialType </ds:X509IssuerSerial> [1]
        <ds:X509SKI> base64Binary </ds:X509SKI> [1]
        <ds:X509SubjectName> string </ds:X509SubjectName> [1]
        <ds:X509Certificate> base64Binary </ds:X509Certificate> [1]
        <ds:X509CRL> base64Binary </ds:X509CRL> [1]
        Allow any elements from a namespace other than this schema's namespace (lax validation). [1]
End Choice
End Sequence
</ds:X509Data>
```

Diagram



Schema Component Representation

```
<element name="X509Data" type=" ds:X509DataType "/>
```

top

Global Definitions

Complex Type: CanonicalizationMethodType

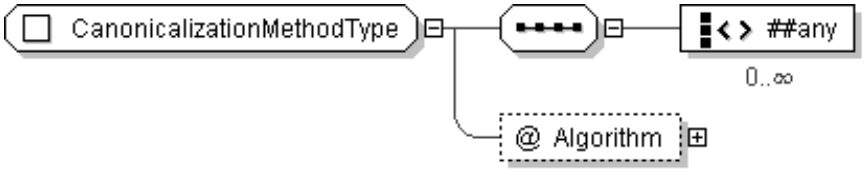
Super-types:	None
Sub-types:	None

Name	CanonicalizationMethodType
Abstract	no

XML Instance Representation

```
<...
Algorithm="anyURI [1]">
<!-- Mixed content -->
   Allow any elements from any namespace (strict validation). [0..*]
</...>
```

Diagram



Schema Component Representation

```
<complexType name="CanonicalizationMethodType" mixed="true">
   <sequence>
      <any namespace="##any" minOccurs="0" maxOccurs="unbounded"/>
      <-- (0,unbounded) elements from (1,1) namespace -->
   </sequence>
   <attribute name="Algorithm" type=" anyURI " use="required"/>
</complexType>
```

top

Complex Type: DSAKeyValueType

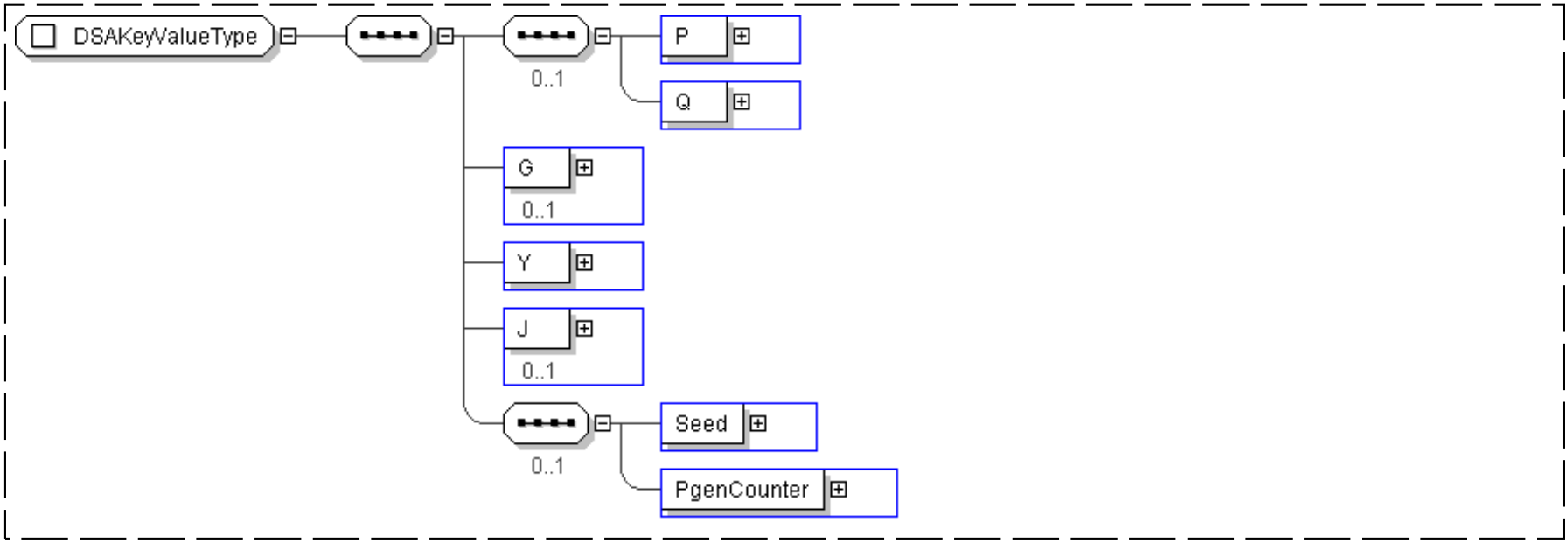
Super-types:	None
Sub-types:	None

Name	DSAKeyValueType
Abstract	no

XML Instance Representation

```
<...>
Start Sequence [0..1]
      <ds:P> ds:CryptoBinary </ds:P> [1]
      <ds:Q> ds:CryptoBinary </ds:Q> [1]
End Sequence
   <ds:G> ds:CryptoBinary </ds:G> [0..1]
   <ds:Y> ds:CryptoBinary </ds:Y> [1]
   <ds:J> ds:CryptoBinary </ds:J> [0..1]
Start Sequence [0..1]
      <ds:Seed> ds:CryptoBinary </ds:Seed> [1]
      <ds:PgenCounter> ds:CryptoBinary </ds:PgenCounter> [1]
End Sequence
</...>
```

Diagram



Schema Component Representation

```
<complexType name="DSAKeyValueType">
   <sequence>
      <sequence minOccurs="0">
         <element name="P" type=" ds:CryptoBinary "/>
         <element name="Q" type=" ds:CryptoBinary "/>
      </sequence>
      <element name="G" type=" ds:CryptoBinary " minOccurs="0"/>
      <element name="Y" type=" ds:CryptoBinary "/>
      <element name="J" type=" ds:CryptoBinary " minOccurs="0"/>
      <sequence minOccurs="0">
         <element name="Seed" type=" ds:CryptoBinary "/>
         <element name="PgenCounter" type=" ds:CryptoBinary "/>
      </sequence>
   </sequence>
</complexType>
```

top

Complex Type: DigestMethodType

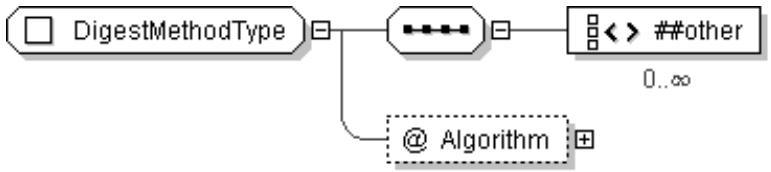
Super-types:	None
Sub-types:	None

Name	DigestMethodType
Abstract	no

XML Instance Representation

```
<...
Algorithm="anyURI [1]">
<!-- Mixed content -->
   Allow any elements from a namespace other than this schema's namespace (lax validation). [0..*]
</...>
```

Diagram



Schema Component Representation

```
<complexType name="DigestMethodType" mixed="true">
   <sequence>
      <any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="Algorithm" type=" anyURI " use="required"/>
</complexType>
```

top

Complex Type: KeyInfoType

Super-types:	None
Sub-types:	None

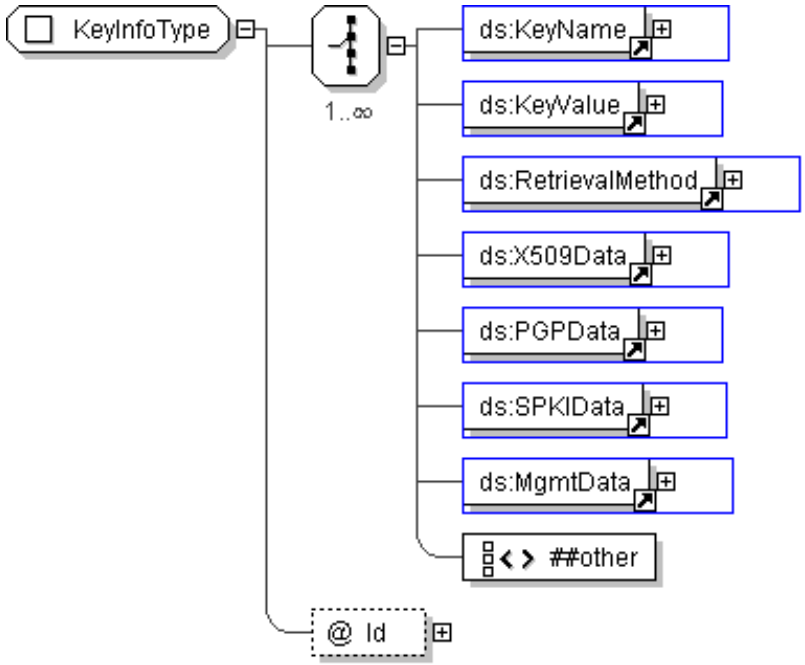
Name	KeyInfoType
Abstract	no

XML Instance Representation

```
<...
Id="ID [0..1]">
<!-- Mixed content -->
```

```
Start Choice [1..*]
      <ds:KeyName> ... </ds:KeyName> [1]
      <ds:KeyValue> ... </ds:KeyValue> [1]
      <ds:RetrievalMethod> ... </ds:RetrievalMethod> [1]
      <ds:X509Data> ... </ds:X509Data> [1]
      <ds:PGPData> ... </ds:PGPData> [1]
      <ds:SPKIData> ... </ds:SPKIData> [1]
      <ds:MgmtData> ... </ds:MgmtData> [1]
      Allow any elements from a namespace other than this schema's namespace (lax validation). [1]
End Choice
</...>
```

Diagram



Schema Component Representation

```
<complexType name="KeyInfoType" mixed="true">
   <choice maxOccurs="unbounded">
      <element ref=" ds:KeyName "/>
      <element ref=" ds:KeyValue "/>
      <element ref=" ds:RetrievalMethod "/>
      <element ref=" ds:X509Data "/>
      <element ref=" ds:PGPData "/>
      <element ref=" ds:SPKIData "/>
```

```
      <element ref=" ds:MgmtData "/>
      <any namespace="##other" processContents="lax"/>
      <-- (1,1) elements from (0,unbounded) namespaces -->
   </choice>
   <attribute name="Id" type=" ID " use="optional"/>
</complexType>
```

top

Complex Type: KeyValueType

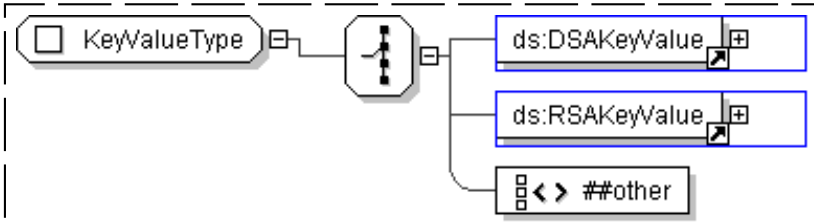
Super-types:	None
Sub-types:	None

Name	KeyValueType
Abstract	no

XML Instance Representation

```
<...>
<!-- Mixed content -->
Start Choice [1]
      <ds:DSAKeyValue> ... </ds:DSAKeyValue> [1]
      <ds:RSAKeyValue> ... </ds:RSAKeyValue> [1]
      Allow any elements from a namespace other than this schema's namespace (lax validation). [1]
End Choice
</...>
```

Diagram



Schema Component Representation

```
<complexType name="KeyValueType" mixed="true">
   <choice>
      <element ref=" ds:DSAKeyValue "/>
      <element ref=" ds:RSAKeyValue "/>
```

```
      <any namespace="##other" processContents="lax"/>
   </choice>
</complexType>
```

top

Complex Type: ManifestType

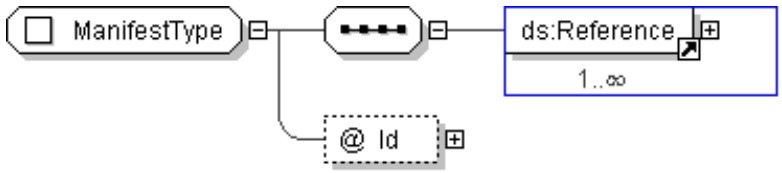
Super-types:	None
Sub-types:	None

Name	ManifestType
Abstract	no

XML Instance Representation

```
<...
Id="ID [0..1]">
   <ds:Reference> ... </ds:Reference> [1..*]
</...>
```

Diagram



Schema Component Representation

```
<complexType name="ManifestType">
   <sequence>
      <element ref=" ds:Reference " maxOccurs="unbounded"/>
   </sequence>
   <attribute name="Id" type=" ID " use="optional"/>
</complexType>
```

top

Complex Type: ObjectType

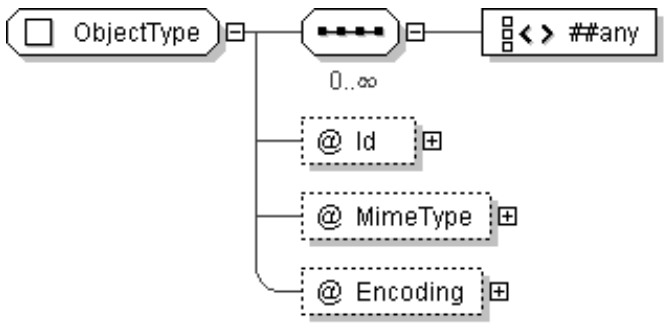
Super-types:	None
Sub-types:	None

Name	ObjectType
Abstract	no

XML Instance Representation

```
<...
Id="ID [0..1]"
MimeType="string [0..1]"
Encoding="anyURI [0..1]">
<!-- Mixed content -->
Start Sequence [0..*]
      Allow any elements from any namespace (lax validation). [1]
End Sequence
</...>
```

Diagram



Schema Component Representation

```
<complexType name="ObjectType" mixed="true">
   <sequence minOccurs="0" maxOccurs="unbounded">
      <any namespace="##any" processContents="lax"/>
   </sequence>
   <attribute name="Id" type=" ID " use="optional"/>
   <attribute name="MimeType" type=" string " use="optional"/>
   <attribute name="Encoding" type=" anyURI " use="optional"/>
   <-- add a grep facet -->
</complexType>
```

top

Complex Type: PGPDataType

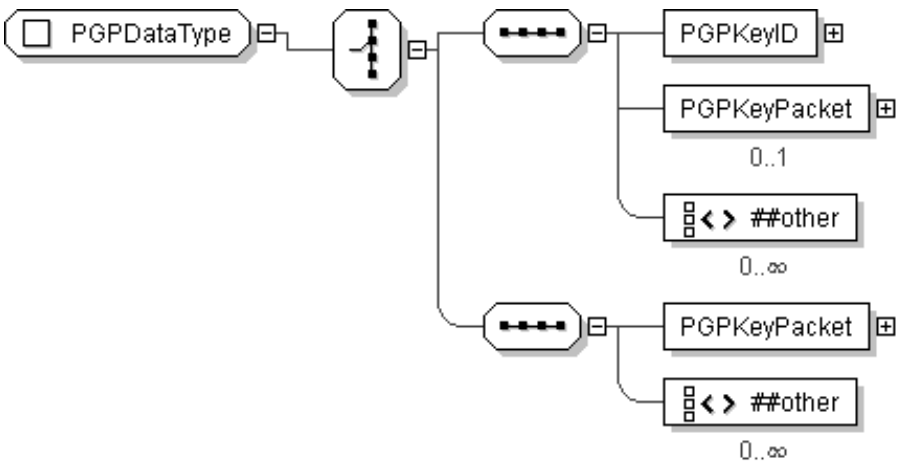
Super-types:	None
Sub-types:	None

Name	PGPDataType
Abstract	no

XML Instance Representation

```
<...>
Start Choice [1]
      <ds:PGPKeyID> base64Binary </ds:PGPKeyID> [1]
      <ds:PGPKeyPacket> base64Binary </ds:PGPKeyPacket> [0..1]
      Allow any elements from a namespace other than this schema's namespace (lax validation). [0..*]
      <ds:PGPKeyPacket> base64Binary </ds:PGPKeyPacket> [1]
      Allow any elements from a namespace other than this schema's namespace (lax validation). [0..*]
End Choice
</...>
```

Diagram



Schema Component Representation

```
<complexType name="PGPDataType">
   <choice>
      <sequence>
         <element name="PGPKeyID" type=" base64Binary "/>
         <element name="PGPKeyPacket" type=" base64Binary " minOccurs="0"/>
         <any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
```

```
      </sequence>
      <sequence>
         <element name="PGPKeyPacket" type=" base64Binary "/>
         <any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
   </choice>
</complexType>
```

top

Complex Type: RSAKeyValueType

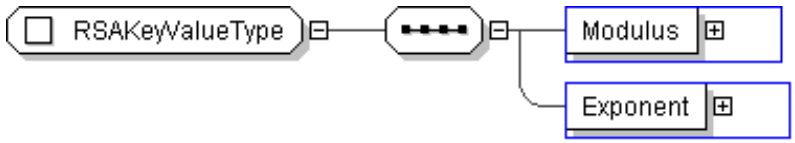
Super-types:	None
Sub-types:	None

Name	RSAKeyValueType
Abstract	no

XML Instance Representation

```
<...>
   <ds:Modulus> ds:CryptoBinary </ds:Modulus> [1]
   <ds:Exponent> ds:CryptoBinary </ds:Exponent> [1]
</...>
```

Diagram



Schema Component Representation

```
<complexType name="RSAKeyValueType">
   <sequence>
      <element name="Modulus" type=" ds:CryptoBinary "/>
      <element name="Exponent" type=" ds:CryptoBinary "/>
   </sequence>
</complexType>
```

top

Complex Type: ReferenceType

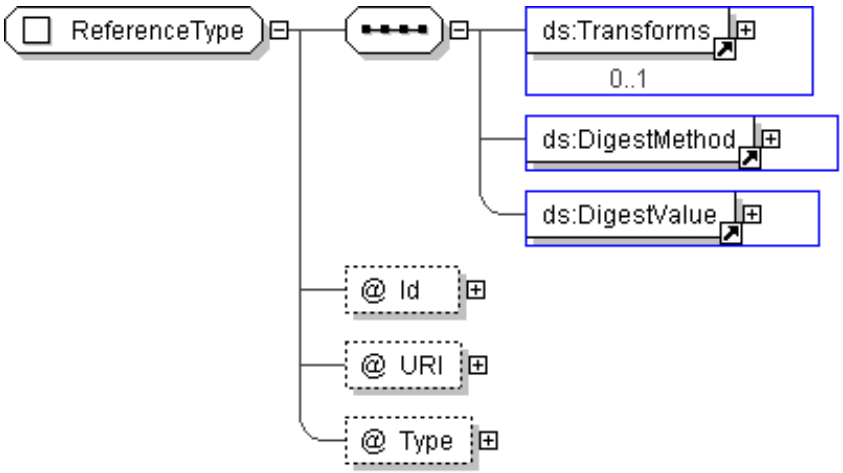
Super-types:	None
Sub-types:	None

Name	ReferenceType
Abstract	no

XML Instance Representation

```
<...
Id="ID [0..1]"
URI="anyURI [0..1]"
Type="anyURI [0..1]">
   <ds:Transforms> ... </ds:Transforms> [0..1]
   <ds:DigestMethod> ... </ds:DigestMethod> [1]
   <ds:DigestValue> ... </ds:DigestValue> [1]
</...>
```

Diagram



Schema Component Representation

```
<complexType name="ReferenceType">
   <sequence>
      <element ref=" ds:Transforms " minOccurs="0"/>
      <element ref=" ds:DigestMethod "/>
      <element ref=" ds:DigestValue "/>
   </sequence>
```

```
   <attribute name="Id" type=" ID " use="optional"/>
   <attribute name="URI" type=" anyURI " use="optional"/>
   <attribute name="Type" type=" anyURI " use="optional"/>
</complexType>
```

top

Complex Type: RetrievalMethodType

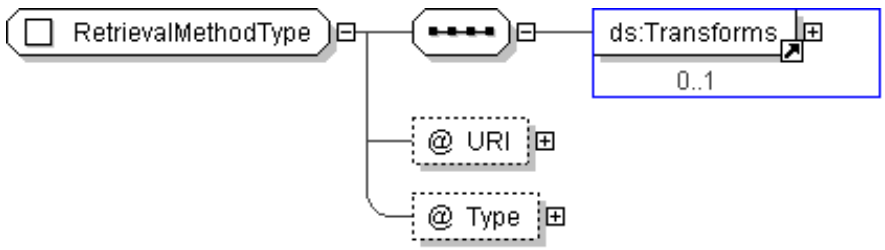
Super-types:	None
Sub-types:	None

Name	RetrievalMethodType
Abstract	no

XML Instance Representation

```
<...
URI="anyURI [0..1]"
Type="anyURI [0..1]">
   <ds:Transforms> ... </ds:Transforms> [0..1]
</...>
```

Diagram



Schema Component Representation

```
<complexType name="RetrievalMethodType">
   <sequence>
      <element ref=" ds:Transforms " minOccurs="0"/>
   </sequence>
   <attribute name="URI" type=" anyURI "/>
   <attribute name="Type" type=" anyURI " use="optional"/>
</complexType>
```

top

Complex Type: SPKIDataType

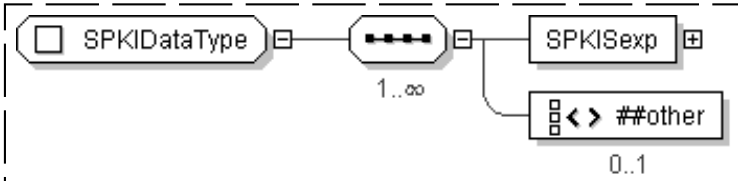
Super-types:	None
Sub-types:	None

Name	SPKIDataType
Abstract	no

XML Instance Representation

```
<...>
Start Sequence [1..*]
      <ds:SPKISexp> base64Binary </ds:SPKISexp> [1]
      Allow any elements from a namespace other than this schema's namespace (lax validation). [0..1]
End Sequence
</...>
```

Diagram



Schema Component Representation

```
<complexType name="SPKIDataType">
   <sequence maxOccurs="unbounded">
      <element name="SPKISexp" type=" base64Binary "/>
      <any namespace="##other" processContents="lax" minOccurs="0"/>
   </sequence>
</complexType>
```

top

Complex Type: SignatureMethodType

Super-types:	None

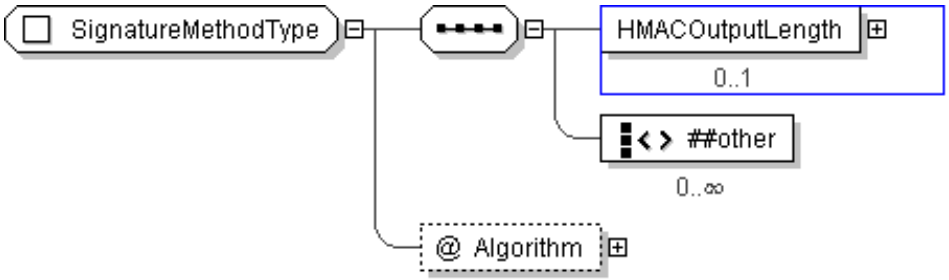
Sub-types:	None

Name	SignatureMethodType
Abstract	no

XML Instance Representation

```
<...
Algorithm="anyURI [1]">
<!-- Mixed content -->
   <ds:HMACOutputLength> ds:HMACOutputLengthType </ds:HMACOutputLength> [0..1]
   Allow any elements from a namespace other than this schema's namespace (strict validation).
   [0..*]
</...>
```

Diagram



Schema Component Representation

```
<complexType name="SignatureMethodType" mixed="true">
   <sequence>
      <element name="HMACOutputLength" type=" ds:HMACOutputLengthType " minOccurs="0"/>
      <any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
      <-- (0,unbounded) elements from (1,1) external namespace -->
   </sequence>
   <attribute name="Algorithm" type=" anyURI " use="required"/>
</complexType>
```

top

Complex Type: SignaturePropertiesType

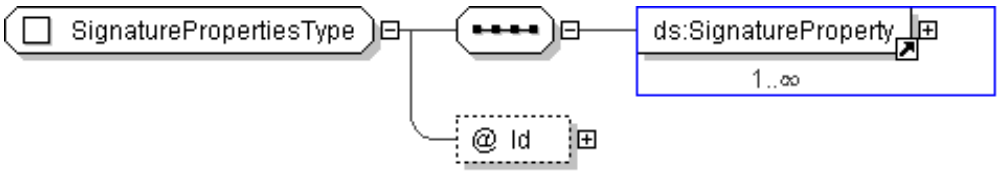
Super-types:	None
Sub-types:	None

Name	SignaturePropertiesType
Abstract	no

XML Instance Representation

```
<...
Id="ID [0..1]">
   <ds:SignatureProperty> ... </ds:SignatureProperty> [1..*]
</...>
```

Diagram



Schema Component Representation

```
<complexType name="SignaturePropertiesType">
   <sequence>
      <element ref=" ds:SignatureProperty " maxOccurs="unbounded"/>
   </sequence>
   <attribute name="Id" type=" ID " use="optional"/>
</complexType>
```

top

Complex Type: SignaturePropertyType

Super-types:	None
Sub-types:	None

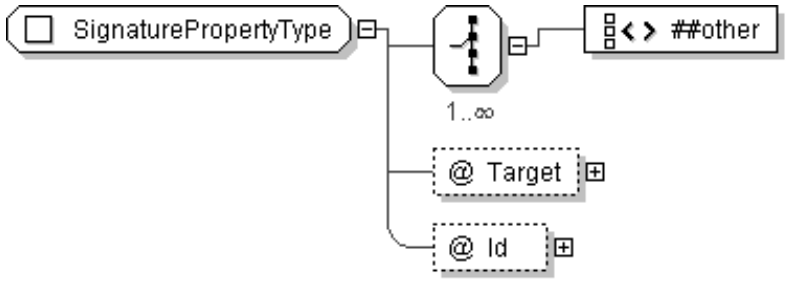
Name	SignaturePropertyType
Abstract	no

XML Instance Representation

```
<...
Target="anyURI [1]"
Id="ID [0..1]">
<!-- Mixed content -->
```

```
Start Choice [1..*]
      Allow any elements from a namespace other than this schema's namespace (lax validation). [1]
End Choice
</...>
```

Diagram



Schema Component Representation

```
<complexType name="SignaturePropertyType" mixed="true">
   <choice maxOccurs="unbounded">
      <any namespace="##other" processContents="lax"/>
      <-- (1,1) elements from (1,unbounded) namespaces -->
   </choice>
   <attribute name="Target" type=" anyURI " use="required"/>
   <attribute name="Id" type=" ID " use="optional"/>
</complexType>
```

top

Complex Type: SignatureType

Super-types:	None
Sub-types:	None

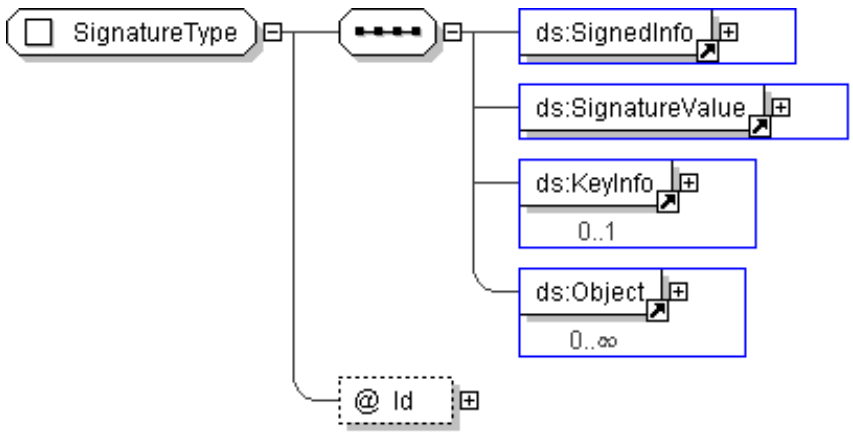
Name	SignatureType
Abstract	no

XML Instance Representation

```
<...
Id="ID [0..1]">
   <ds:SignedInfo> ... </ds:SignedInfo> [1]
   <ds:SignatureValue> ... </ds:SignatureValue> [1]
```

```
   <ds:KeyInfo> ... </ds:KeyInfo> [0..1]
   <ds:Object> ... </ds:Object> [0..*]
</...>
```

Diagram



Schema Component Representation

```
<complexType name="SignatureType">
   <sequence>
      <element ref=" ds:SignedInfo "/>
      <element ref=" ds:SignatureValue "/>
      <element ref=" ds:KeyInfo " minOccurs="0"/>
      <element ref=" ds:Object " minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="Id" type=" ID " use="optional"/>
</complexType>
```

top

Complex Type: SignatureValueType

Super-types:	base64Binary < **SignatureValueType** (by extension)
Sub-types:	None

Name	SignatureValueType
Abstract	no

XML Instance Representation

```
<...
Id="ID [0..1]">
base64Binary
</...>
```

Diagram

Schema Component Representation

```
<complexType name="SignatureValueType">
   <simpleContent>
      <extension base=" base64Binary ">
         <attribute name="Id" type=" ID " use="optional"/>
      </extension>
   </simpleContent>
</complexType>
```

top

Complex Type: SignedInfoType

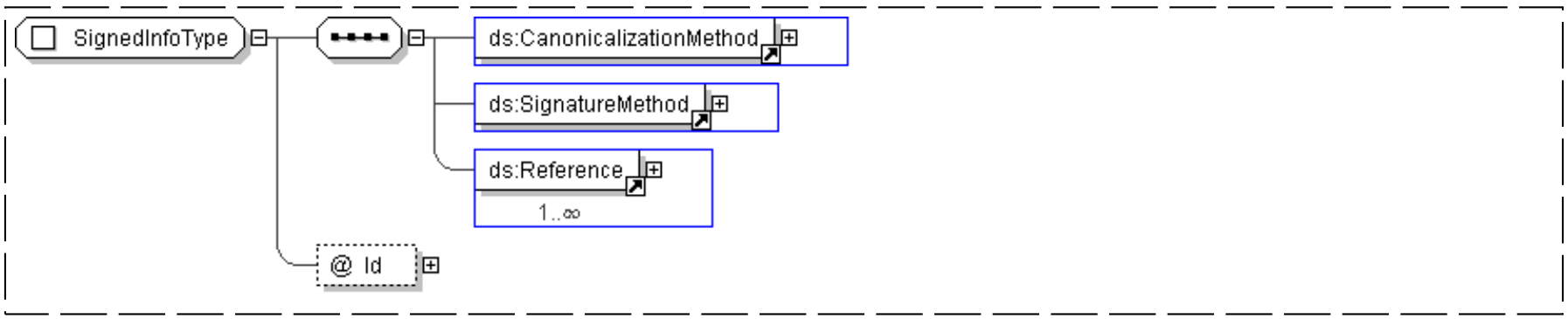
Super-types:	None
Sub-types:	None

Name	SignedInfoType
Abstract	no

XML Instance Representation

```
<...
Id="ID [0..1]">
   <ds:CanonicalizationMethod> ... </ds:CanonicalizationMethod> [1]
   <ds:SignatureMethod> ... </ds:SignatureMethod> [1]
   <ds:Reference> ... </ds:Reference> [1..*]
</...>
```

Diagram



Schema Component Representation

```
<complexType name="SignedInfoType">
   <sequence>
      <element ref=" ds:CanonicalizationMethod "/>
      <element ref=" ds:SignatureMethod "/>
      <element ref=" ds:Reference " maxOccurs="unbounded"/>
   </sequence>
   <attribute name="Id" type=" ID " use="optional"/>
</complexType>
```

top

Complex Type: TransformType

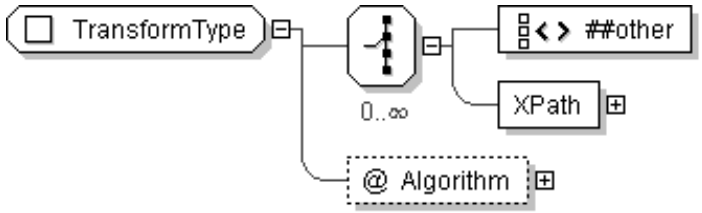
Super-types:	None
Sub-types:	None

Name	TransformType
Abstract	no

XML Instance Representation

```
<...
Algorithm="anyURI [1]">
<!-- Mixed content -->
Start Choice [0..*]
      Allow any elements from a namespace other than this schema's namespace (lax validation). [1]
      <ds:XPath> string </ds:XPath> [1]
End Choice
</...>
```

Diagram



Schema Component Representation

```
<complexType name="TransformType" mixed="true">
   <choice minOccurs="0" maxOccurs="unbounded">
      <any namespace="##other" processContents="lax"/>
      <-- (1,1) elements from (0,unbounded) namespaces -->
      <element name="XPath" type=" string "/>
   </choice>
   <attribute name="Algorithm" type=" anyURI " use="required"/>
</complexType>
```

top

Complex Type: TransformsType

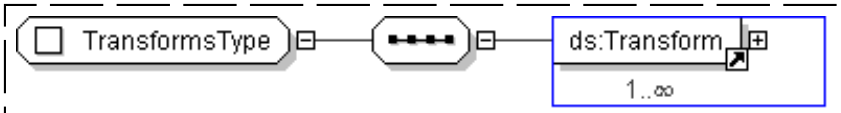
Super-types:	None
Sub-types:	None

Name	TransformsType
Abstract	no

XML Instance Representation

```
<...>
   <ds:Transform> ... </ds:Transform> [1..*]
</...>
```

Diagram



Schema Component Representation

```
<complexType name="TransformsType">
   <sequence>
      <element ref=" ds:Transform " maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

top

Complex Type: X509DataType

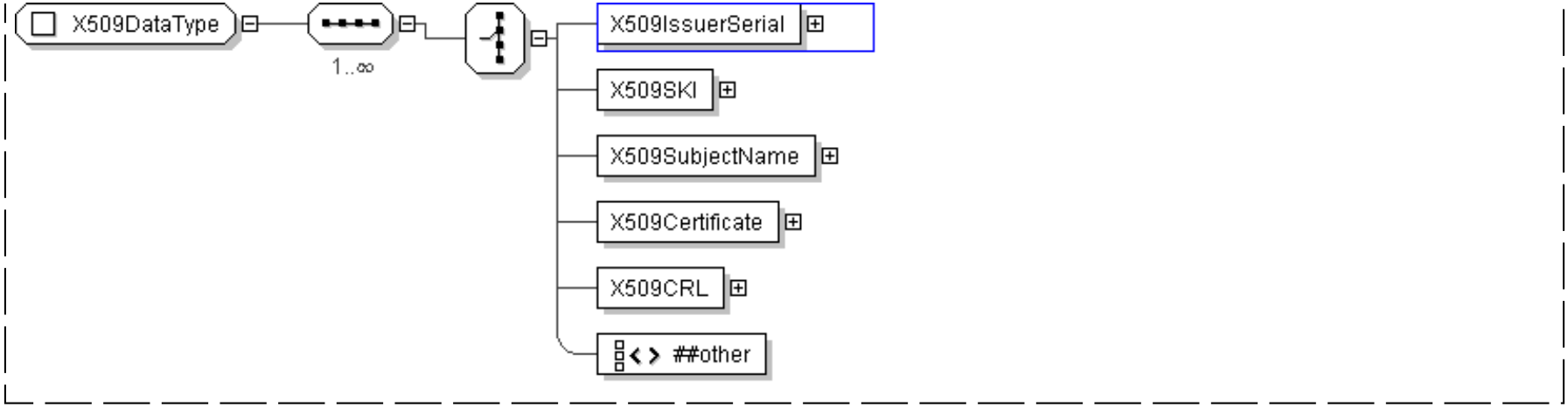
Super-types:	None
Sub-types:	None

Name	X509DataType
Abstract	no

XML Instance Representation

```
<...>
Start Sequence [1..*]
Start Choice [1]
         <ds:X509IssuerSerial> ds:X509IssuerSerialType </ds:X509IssuerSerial> [1]
         <ds:X509SKI> base64Binary </ds:X509SKI> [1]
         <ds:X509SubjectName> string </ds:X509SubjectName> [1]
         <ds:X509Certificate> base64Binary </ds:X509Certificate> [1]
         <ds:X509CRL> base64Binary </ds:X509CRL> [1]
         Allow any elements from a namespace other than this schema's namespace (lax validation). [1]
End Choice
End Sequence
</...>
```

Diagram



Schema Component Representation

```
<complexType name="X509DataType">
   <sequence maxOccurs="unbounded">
      <choice>
         <element name="X509IssuerSerial" type=" ds:X509IssuerSerialType "/>
         <element name="X509SKI" type=" base64Binary "/>
         <element name="X509SubjectName" type=" string "/>
         <element name="X509Certificate" type=" base64Binary "/>
         <element name="X509CRL" type=" base64Binary "/>
         <any namespace="##other" processContents="lax"/>
      </choice>
   </sequence>
</complexType>
```

top

Complex Type: X509IssuerSerialType

Super-types:	None
Sub-types:	None

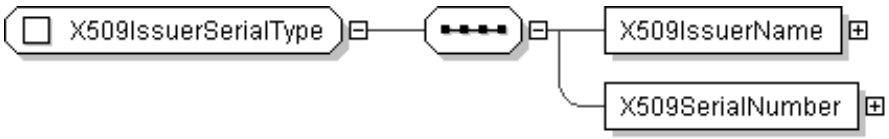
Name	X509IssuerSerialType
Abstract	no

XML Instance Representation

```
<...>
   <ds:X509IssuerName> string </ds:X509IssuerName> [1]
```

```
   <ds:X509SerialNumber> integer </ds:X509SerialNumber> [1]
</...>
```

Diagram



Schema Component Representation

```
<complexType name="X509IssuerSerialType">
   <sequence>
      <element name="X509IssuerName" type=" string "/>
      <element name="X509SerialNumber" type=" integer "/>
   </sequence>
</complexType>
```

top

Simple Type: CryptoBinary

Super-types:	base64Binary < **CryptoBinary** (by restriction)
Sub-types:	None

Name	CryptoBinary
Content	• Base XSD Type: base64Binary

Diagram

Schema Component Representation

```
<simpleType name="CryptoBinary">
   <restriction base=" base64Binary "/>
</simpleType>
```

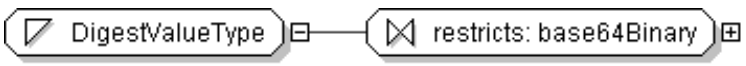
top

Simple Type: DigestValueType

Super-types:	base64Binary < **DigestValueType** (by restriction)
Sub-types:	None

Name	DigestValueType
Content	• Base XSD Type: base64Binary

Diagram



Schema Component Representation

```
<simpleType name="DigestValueType">
   <restriction base=" base64Binary "/>
</simpleType>
```

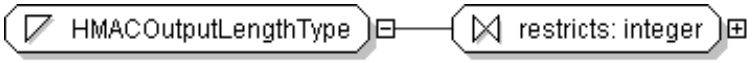
top

Simple Type: HMACOutputLengthType

Super-types:	integer < **HMACOutputLengthType** (by restriction)
Sub-types:	None

Name	HMACOutputLengthType
Content	• Base XSD Type: integer

Diagram



Schema Component Representation

```
<simpleType name="HMACOutputLengthType">
   <restriction base=" integer "/>
</simpleType>
```

top

Legend

Complex Type: AusAddress

Schema Component Type — Schema Component Name

Super-types:	Address < AusAddress (by extension)
Sub-types:	• QLDAddress (by restriction)

If this schema component is a type definition, its type hierarchy is shown in a gray-bordered box.

Name	AusAddress
Abstract	no

The table above displays the properties of this schema component.

XML Instance Representation

```
<... country="Australia" >
<unitNo> string </unitNo> [0..1]
<houseNo> string </houseNo> [1]
<street> string </street> [1]
Start Choice [1]
<city> string </city> [1]
<town> string </town> [1]
End Choice
<state> AusStates </state> [1]
<postcode> string <<pattern = [1-9][0-9]{3}>> </postcode> [1]
</...>
```

The XML Instance Representation table above shows the schema component's content as an XML instance.

- The minimum and maximum occurrence of elements and attributes are provided in square brackets, e.g. [0..1].
- Model group information are shown in gray, e.g. Start Choice ... End Choice.
- For type derivations, the elements and attributes that have been added to or changed from the base type's content are shown in **bold**.
- If an element/attribute has a fixed value, the fixed value is shown in green, e.g. country="Australia".
- Otherwise, the type of the element/attribute is displayed.
  - If the element/attribute's type is in the schema, a link is provided to it.
  - For local simple type definitions, the constraints are displayed in angle brackets, e.g. <<*pattern* = [1-9][0-9]{3}>>.

Schema Component Representation

```
<complexType name="AusAddress">
<complexContent>
<extension base=" Address ">
<sequence>
<element name="state" type=" AusStates "/>
<element name="postcode">
<simpleType>
<restriction base=" string ">
<pattern value="[1-9][0-9]{3}"/>
</restriction>
</simpleType>
</element>
</sequence>
<attribute name="country" type=" string " fixed="Australia"/>
</extension>
</complexContent>
</complexType>
```

The Schema Component Representation table above displays the underlying XML representation of the schema component. (Annotations are not shown.)

top

Glossary

Abstract (Applies to complex type definitions and element declarations). An abstract element or complex type cannot used to validate an element instance. If there is a reference to an abstract element, only element declarations that can substitute the abstract element can be used to validate the instance. For references to abstract type definitions, only derived types can be used.

All Model Group Child elements can be provided *in any order* in instances. See: http://www.w3.org/TR/xmlschema-1/#element-all.

Choice Model Group *Only one* from the list of child elements and model groups can be provided in instances. See: http://www.w3.org/TR/xmlschema-1/#element-choice.

Collapse Whitespace Policy Replace tab, line feed, and carriage return characters with space character (Unicode character 32). Then, collapse contiguous sequences of space characters into single space character, and remove leading and trailing space characters.

Disallowed Substitutions (Applies to element declarations). If *substitution* is specified, then substitution group members cannot be used in place of the given element declaration to validate element instances. If *derivation methods*, e.g. extension, restriction, are specified, then the given element declaration will not validate element instances that have types derived from the element declaration's type using the specified derivation methods. Normally, element instances can override their declaration's type by specifying an `xsi:type` attribute.

Key Constraint Like Uniqueness Constraint, but additionally requires that the specified value(s) must be provided. See: http://www.w3.org/TR/xmlschema-1/#cIdentity-constraint_Definitions.

Key Reference Constraint Ensures that the specified value(s) must match value(s) from a Key Constraint or Uniqueness Constraint. See: http://www.w3.org/TR/xmlschema-1/#cIdentity-constraint_Definitions.

Model Group Groups together element content, specifying the order in which the element content can occur and the number of times the group of element content may be repeated. See: http://www.w3.org/TR/xmlschema-1/#Model_Groups.

Nillable (Applies to element declarations). If an element declaration is nillable, instances can use the `xsi:nil` attribute. The `xsi:nil` attribute is the boolean attribute, *nil*, from the *http://www.w3.org/2001/XMLSchema-instance* namespace. If an element instance has an `xsi:nil` attribute set to true, it can be left empty, even though its element declaration may have required content.

Notation A notation is used to identify the format of a piece of data. Values of elements and attributes that are of type, NOTATION, must come from the names of declared notations. See: http://www.w3.org/TR/xmlschema-1/#cNotation_Declarations.

Preserve Whitespace Policy Preserve whitespaces exactly as they appear in instances.

Prohibited Derivations (Applies to type definitions). Derivation methods that cannot be used to create sub-types from a given type definition.

Prohibited Substitutions (Applies to complex type definitions). Prevents sub-types that have been derived using the specified derivation methods from validating element instances in place of the given type definition.

Replace Whitespace Policy Replace tab, line feed, and carriage return characters with space character (Unicode character 32).

Sequence Model Group Child elements and model groups must be provided *in the specified order* in instances. See: http://www.w3.org/TR/xmlschema-1/#element-sequence.

Substitution Group Elements that are *members* of a substitution group can be used wherever the *head* element of the substitution group is referenced.

Substitution Group Exclusions (Applies to element declarations). Prohibits element declarations from nominating themselves as being able to substitute a given element declaration, if they have types that are derived from the original element's type using the specified derivation methods.

Target Namespace The target namespace identifies the namespace that components in this schema belongs to. If no target namespace is provided, then the schema components do not belong to any namespace.

Uniqueness Constraint Ensures uniqueness of an element/attribute value, or a combination of values, within a specified scope. See: http://www.w3.org/TR/xmlschema-1/#cIdentity-constraint_Definitions.

top

Generated by <oXygen/> XML Editor using a modified version of xs3p that adds schema diagrams and chunking support.