XMLmodeling.com

UML Profile for XML Schema

This version:

March 3, 2008

Latest version:

http://www.xmlmodeling.com/documentation/specs/

Authors:

David Carlson <dcarlson@xmlmodeling.com>

See Also:

UML Profile for XSD Generation, http://www.xmlmodeling.com/documentation/specs/

Copyright © 2001-2008 David A Carlson, All rights reserved.

This specification is made available under the terms of the Eclipse Public License v1.0 which accompanies this distribution, and is available at http://www.eclipse.org/legal/epl-v10.html

Abstract:

This profile includes stereotypes and properties that are required to support reverse-engineering of XML Schema definitions into UML models and generation of schemas from UML. The profile is compliant with the UML 2.1 specification. UML constructs are reused when they have the same semantics or when an acceptable default mapping is possible from a UML construct to a corresponding XML Schema component. An important factor in this profile's design is that stereotypes must be applied to a UML model only when the default mapping is insufficient.

Default Mapping from UML

Prior to defining the stereotypes and their semantics, it is helpful to describe the default mapping between an unadorned UML model and XML Schema. With this background, each stereotype can be defined by how it modifies the default mapping or adds metadata that define additional characteristics of XSD components.

Many of the mappings can be illustrated using XML Schemas that were developed as part of the Financial Product Markup Language (FpML) specification (see www.fpml.org). All of the examples used in this profile specification are borrowed directly from FpML, unless otherwise noted.

Package

Each UML package is mapped to a separate XML Schema file, using the default null targetNamespace. `xsd:include` declarations are added for any types that are referenced in other packages. The hierarchical structure of the UML nested packages is replicated in a file folder hierarchy that contains the generated schemas, and `xsd:include` statements use relative path schema locations in this folder hierarchy.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      attributeFormDefault="unqualified" elementFormDefault="qualified">
  <xsd:include schemaLocation="fpml-shared-4-2.xsd"/>
  ...
</xsd:schema>
```

Comment

Each UML comment is mapped to an <xsd:documentation> tag that is contained within the schema component corresponding to the mapped owner of the UML comment.

For example, comments owned by an Enumeration are mapped to documentation in a simpleType definition, and comments owned by contained EnumerationLiteral are mapped to schema enumeration facets.

```
  <xsd:simpleType name="DayTypeEnum">
    <xsd:annotation>
      <xsd:documentation>A day type classification used in counting the
            number of days between two dates.</xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:token">
      <xsd:enumeration value="Business">
        <xsd:annotation>
          <xsd:documentation>When calculating the number of days between two
            dates the count includes only business days.</xsd:documentation>
        </xsd:annotation>
      </xsd:enumeration>

      ...

  </xsd:simpleType>
```

Class

Each UML class is mapped to an XML Schema complexType definition, using the class name as the complexType name. If necessary, the name is adjusted to remove invalid XML name characters such as space or colon.

The complexType is defined using a sequence content model.

A global element declaration is created for each globally defined complexType.

```
<xsd:element name="Commission" type="Commission"/>

<xsd:complexType name="Commission">
  <xsd:sequence>

      ... element content

  </xsd:sequence>
</xsd:complexType>
```

Generalization

Generalization between two UML Classes is mapped to complexType extension.

Generalization between two UML DataTypes is mapped to simpleType restriction.

```
<xsd:complexType name="MutualFund">
  <xsd:complexContent>
    <xsd:extension base="UnderlyingAsset">
      <xsd:sequence>

      ...

</xsd:complexType>
```

Property

Each UML Property owned by a Class is mapped to a local element declaration in the corresponding schema complexType.

```
<xsd:complexType name="Commission">
  <xsd:sequence>
    <xsd:element name="commissionDenomination"
                 type="CommissionDenominationEnum"/>
    <xsd:element name="commissionAmount" type="xsd:decimal"/>
    <xsd:element name="currency" type="xsd:double" minOccurs="0"/>
    <xsd:element name="commissionPerTrade" type="xsd:decimal"
                 minOccurs="0"/>
    <xsd:element name="fxRate" type="FxRate"
                 minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

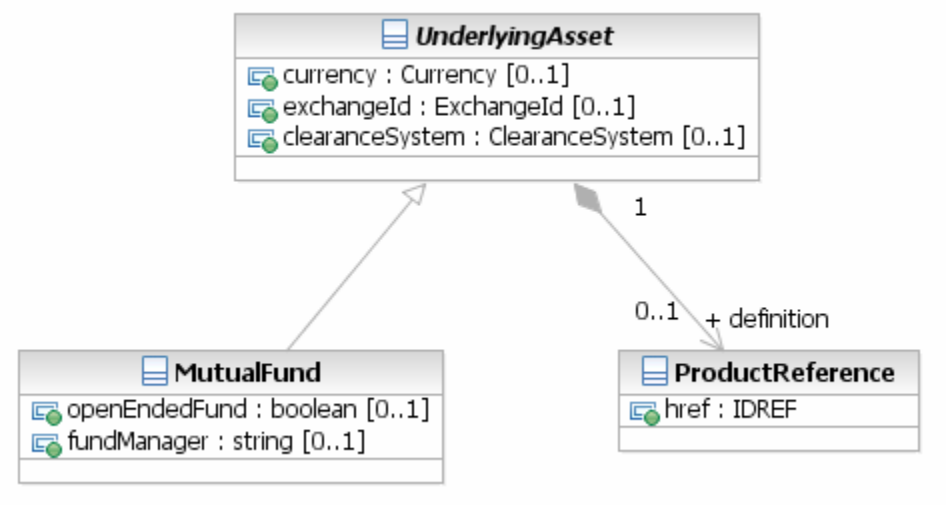
If a UML property has a default value, then it is mapped to the schema element default value constraint.

If a UML property has the `readOnly` attribute set to `true`, and there is a default value, then the schema element has a `fixed` value constraint.

Association

In UML, a navigable property may be owned by the source Class, or owned by the Association and marked as navigable. Navigable properties owned by the Association are omitted from the default schema mapping, because there is no information about their position within the complexType sequence.

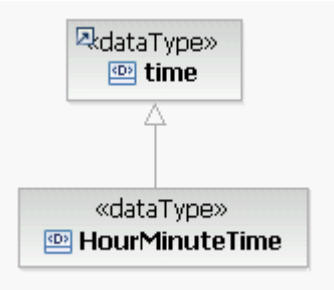
The following example from FpML illustrates the default mapping of classes, properties, associations, and generalization. The recommended convention for class diagrams is that properties with DataType types are shown in the class attributes compartment, and properties with Class types are shown using associations. In this model, Currency, ExchangeId, and ClearanceSystem are represented using UML DataType.




```
<xsd:complexType abstract="true" name="UnderlyingAsset">
  <xsd:sequence>
    <xsd:element name="currency" type="fpml:Currency" minOccurs="0"/>
    <xsd:element name="exchangeId" type="fpml:ExchangeId" minOccurs="0" />
    <xsd:element name="clearanceSystem" type="fpml:ClearanceSystem"
                 minOccurs="0" />
    <xsd:element name="definition" type="fpml:ProductReference"
                 minOccurs="0" />
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="MutualFund">
  <xsd:complexContent>
    <xsd:extension base="fpml:UnderlyingAsset">
      <xsd:sequence>
        <xsd:element name="openEndedFund" type="xsd:boolean"
                     minOccurs="0" />
        <xsd:element name="fundManager" type="xsd:string" minOccurs="0" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="ProductReference">
  <xsd:sequence>
    <xsd:element name="href" type="xsd:IDREF"/>
  </xsd:sequence>
</xsd:complexType>
```

DataType or PrimitiveType

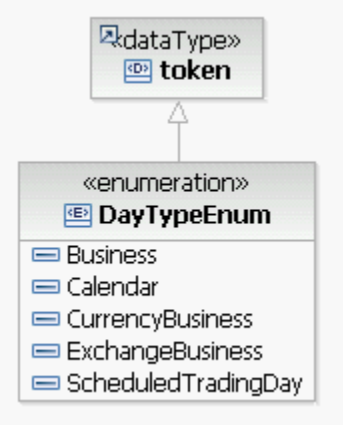
Both UML DataType and PrimitiveType are mapped to schema simpleType definitions. A UML generalization to another data type is represented as a restriction in the schema. If no generalization is present in the UML model, then the simpleType should restrict `xsd:string`.




```
<xsd:simpleType name="HourMinuteTime">
  <xsd:restriction base="xsd:time" />
</xsd:simpleType>
```

Enumeration

An enumeration is a kind of UML DataType, so the default mapping to schema simpleType applied in this case also. All EnumerationLiteral members of the enumeration are mapped to facets in the schema definition.

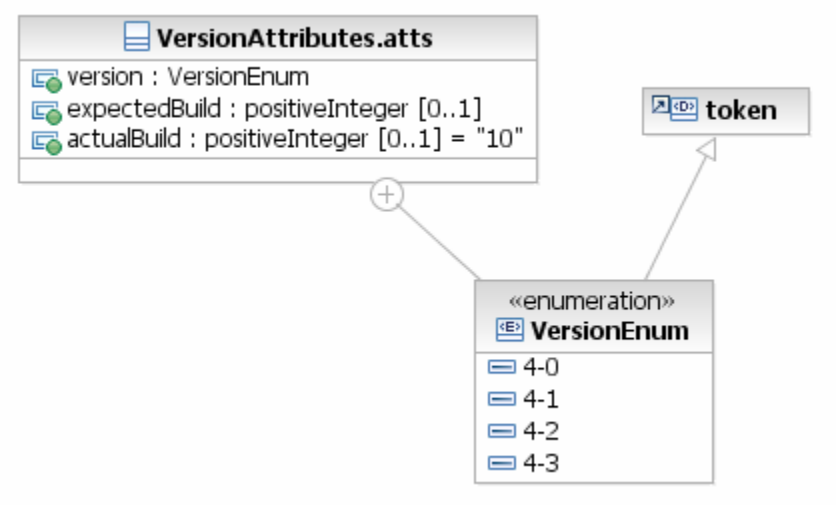



```
<xsd:simpleType name="DayTypeEnum">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="Business"/>
    <xsd:enumeration value="Calendar"/>
    <xsd:enumeration value="CurrencyBusiness"/>
    <xsd:enumeration value="ExchangeBusiness"/>
    <xsd:enumeration value="ScheduledTradingDay"/>
  </xsd:restriction>
</xsd:simpleType>
```

Nested Class or DataType

If a UML Class or DataType is nested within another class, and the nested classifier is referenced as the type of a property contained in the same parent class, then this nested classifier will be mapped to an anonymous type in the schema. The nested classifier is mapped as described above for Class, DataType, or Enumeration, except that it is anonymous within the schema feature where it is used.

This example is taken from FpML, although two stereotypes are omitted that are required for an accurate mapping to that schema. The `VersionAttributes.atts` class should be mapped to an XSD attribute group containing attributes. The default mapping does yield a valid schema, but it is not precisely what is needed by the FpML designers. See the <<attributeGroup>> stereotype definition for a revised example.

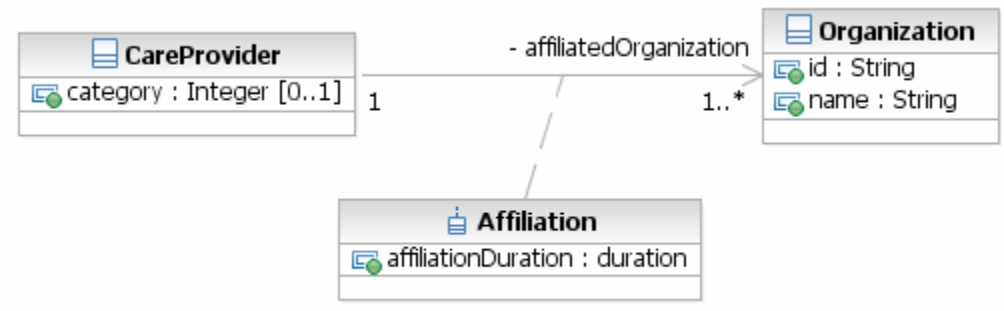



```
<xsd:complexType name="VersionAttributes.atts">
  <xsd:sequence>
    <xsd:element name="version">
      <xsd:simpleType>
        <xsd:restriction base="xsd:token">
          <xsd:enumeration value="4-0"/>
          <xsd:enumeration value="4-1"/>
          <xsd:enumeration value="4-2"/>
          <xsd:enumeration value="4-3"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="expectedBuild" type="xsd:positiveInteger"
                 minOccurs="0"/>
    <xsd:element name="actualBuild" type="xsd:positiveInteger"
                 fixed="10" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

AssociationClass

The default mapping for a UML AssociationClass is supported for transformation from UML to XML schema, but not on mapping from a schema to UML.

The following example (not from FpML) illustrates this mapping.



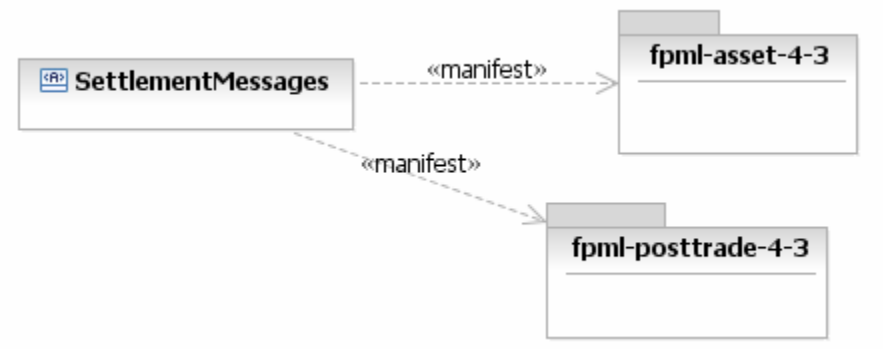
```
<xsd:complexType name="CareProvider">
  <xsd:sequence>
    <xsd:element name="category" type="xsd:int" minOccurs="0" />
    <xsd:element name="affiliation" type="Affiliation"
                 maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Affiliation">
  <xsd:sequence>
    <xsd:element name="affiliationDuration" type="xsd:duration"/>
    <xsd:element name="affiliatedOrganization" type="Organization"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Organization">
  <xsd:sequence>
    <xsd:element name="id" type="xsd:string"/>
    <xsd:element name="name" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

Artifact

The default mapping includes a one-to-one correspondence between each UML package and an XML schema file, where the schema file name is equal to the package name with a ".xsd" file extension. But it is sometimes necessary to generate XSD artifacts that are not aligned with packages. This can be done using a UML Artifact.

A UML Artifact must have one or more Manifestation relationships to packages that are used to create one XML schema file. The schema will include all definitions contained in the manifested package(s). The schema file name will be equal to the Artifact name with ".xsd" extension, or the optional `fileName` property of the UML Artifact may be used to specify an explicit name with a relative or absolute file path.

In this example, the `SettlementMessages` artifact generates one XML schema that contains all definitions from two model packages.



PackageImport

The default mapping calculates all `xsd:include` directives that are required for schema types referenced in other UML model packages and their corresponding schema files. It is not necessary to define explicit package dependencies in the UML model.

However, it is sometimes necessary to design an XML schema that contains no type definitions, but the schema represents a convenient aggregation of several other schemas that are used in combination. This example is drawn from the FIXML financial standard (www.fixprotocol.org). When a UML PackageImport relationship is created between two packages, then the generated schema for the client package includes the explicitly modeled `xsd:include` directive.

The `fixml-order-impl.xsd` file contains this schema, and no type definitions:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:include schemaLocation="fixml-order-base.xsd" />
<xsd:schema>
```

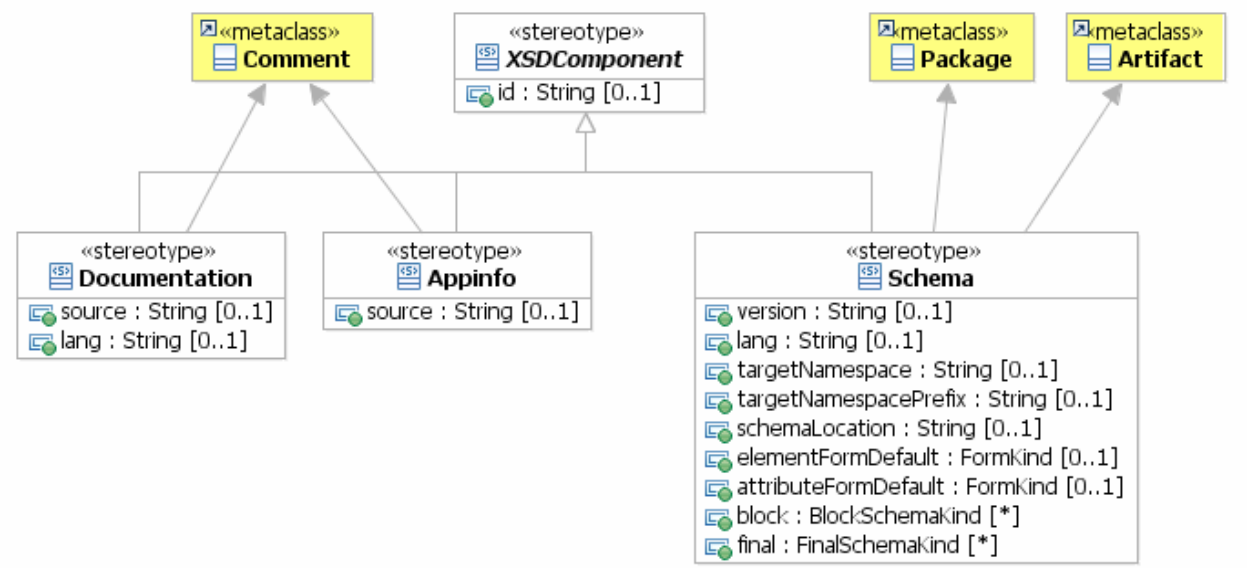
The `fixml-order-impl.xsd` file contains schema type definitions and `xsd:include` directives for other schema files. The `xsd:include` directives in this second file are calculated automatically from the UML model relationships.

Stereotypes Definitions

In general, you should apply stereotypes to the UML model only when you must modify the default mapping from UML to XML Schema, or when you must specify additional XSD attributes that have no corresponding interpretation in UML (e.g. the targetNamespace URI for a schema).

A stereotype icon, if defined, is shown in parentheses following the stereotype name.

Package and Comment



«XSDComponent»

Description

Abstract stereotype that defines properties inherited by XSD components.

Properties

- id :String [0..1] Identifier attribute assigned to an XML Schema component.

Notation and Use

```
<xsd:simpleType id="text" name="TextType">
      ...
</xsd:simpleType>
```

«schema»

Stereotype: Schema

Base Metaclass

- Package
- Artifact

Generalizations

- XSDComponent

Description

In an XML Schema document, the root element <xsd:schema> contains definitions for one target namespace. This stereotype and its assigned property values apply to all owned packages and types, unless a stereotype is assigned to an owned member where it may override properties from the owner package stereotype.

Properties

- version : String [0..1]
  The version of this schema.
- lang : String [0..1]
  The xml:lang attribute in a schema definition.
- targetNamespace : String [0..1]
  A URI representing a globally unique XML namespace that will contain this schema's definitions.
- targetNamespacePrefix : String [0..1]
  The prefix associated with the target XML namespace.
- elementFormDefault : FormKind [0..1]
- attributeFormDefault : FormKind [0..1]
- block : BlockSchemaKind [0..*]
- final : FinalSchemaKind [0..*]

Constraints

-

Semantics

Notation and Use

«documentation» ( )

Stereotype: Documentation

Base Metaclass

- Comment

Generalizations

- XSDComponent

Description

Applied to a UML Comment when the supplementary stereotype properties are needed.

Properties

- source : String [0..1]

- lang : String [0..1]

Constraints

- Only one of «documentation» or «appinfo» may be applied.

Semantics

Notation and Use

This example is from an FpML schema:

```
  <xsd:complexType name="PriceQuoteUnits">
    <xsd:annotation>
      <xsd:documentation source="http://www.FpML.org" xml:lang="en">The units
in which a price is quoted.</xsd:documentation>
    </xsd:annotation>
```

«appinfo» ( )

Stereotype: Appinfo

Base Metaclass

- Comment

Generalizations

- XSDComponent

Description

Must be applied to a UML Comment that is mapped to an XSD appinfo annotation.

Properties

- source : String [0..1]

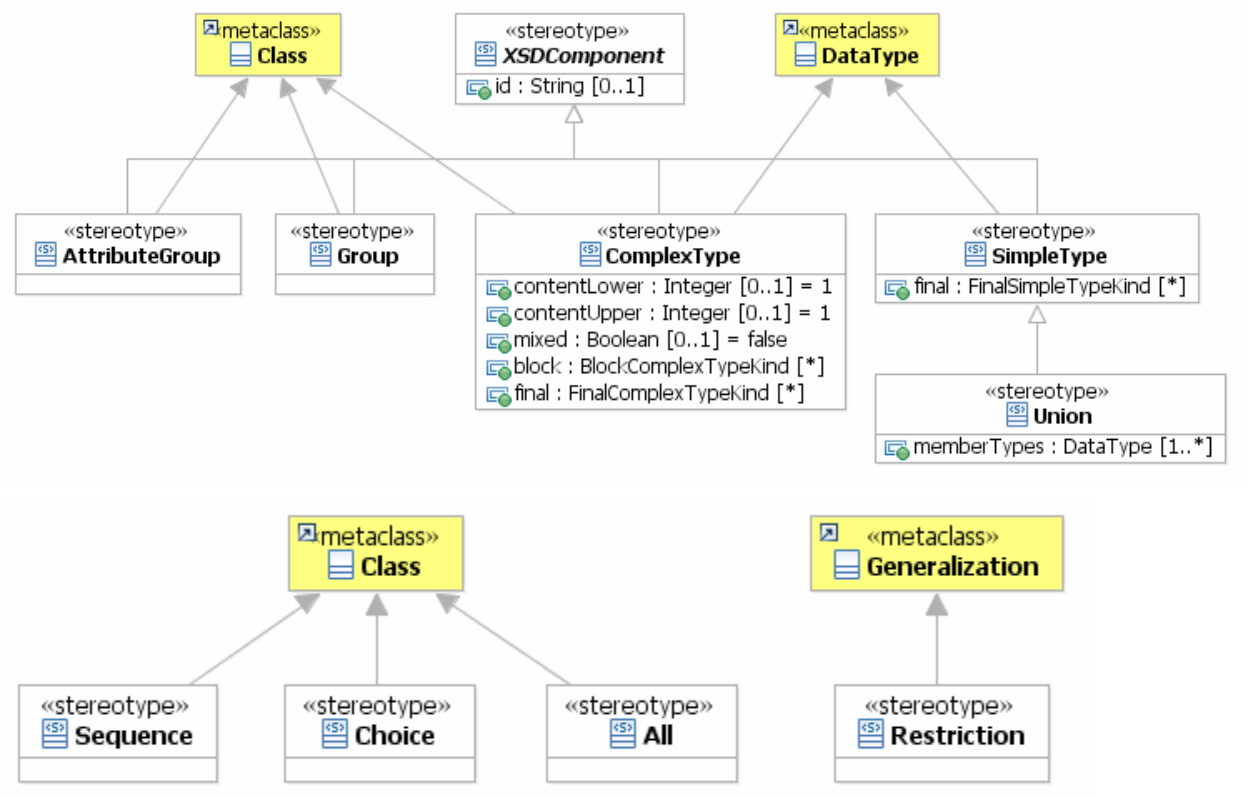
Constraints

- Only one of «documentation» or «appinfo» may be applied.

Semantics

Notation and Use

Class, DataType and Generalization



«complexType» ( )

Stereotype: ComplexType

Base Metaclass

- Class
- DataType

Generalizations

- XSDComponent

Description

This stereotype is rarely applied to a Class because its default mapping is to a schema complexType, but must be applied when the supplementary information from the stereotype properties is required.

This stereotype must be applied to a DataType when it represents a complexType with simpleContent in XML Schema.

Properties

- mixed : Boolean [0..1] = false
- block : BlockComplexTypeKind [0..*]
- final : FinalComplexTypeKind [0..*]
- contentLower : Integer [0..1] = 1
- contentUpper : Integer [0..1] = 1

Constraints

- Shall not be applied in combination with «group», «attributeGroup», «simpleType», «list», or «union».

Semantics

Notation and Use

«restriction»

Stereotype: Restriction

Base Metaclass

- Generalization

Generalizations

-

Description

This stereotype may be applied to a generalization relationship between two Classes or between two DataTypes to signify that an XML Schema restriction is required, instead of an extension. In practice, it is not necessary to assign this stereotype to a generalization between two DataTypes to represent simpleType restriction, because XML Schema extension is invalid in that situation.

Properties

- none

Constraints

- Both general and special classifiers must represent an XML Schema complexType (by default mapping or by stereotype application), or both must represent a simpleType.

Semantics

Notation and Use

«simpleType» ( )

Stereotype: SimpleType

Base Metaclass

- DataType

Generalizations

- XSDComponent

Description

This stereotype may be applied to a DataType that represents an XML Schema simpleType definition. In practice, the stereotype application is only required in the case where the final property must be specified for mapping to the corresponding simpleType attribute. Without a stereotype, a UML DataType will be mapped to XSD simpleType by default.

A DataType that contains a binding to the List template provided by this profile will be mapped to a simpleType list, instead of to an atomic datatype. DataTypes with the List template binding may be displayed using an alternative stereotype icon:

A DataType with private visibility is mapped to an anonymous simpleType definition in schema elements or attributes that use the private type. This mapping is required when an anonymous simpleType is required on a DataType attribute because the UML specification does not allow nested classifiers in a DataType.

Properties

- final : FinalSimpleTypeKind [0..*]

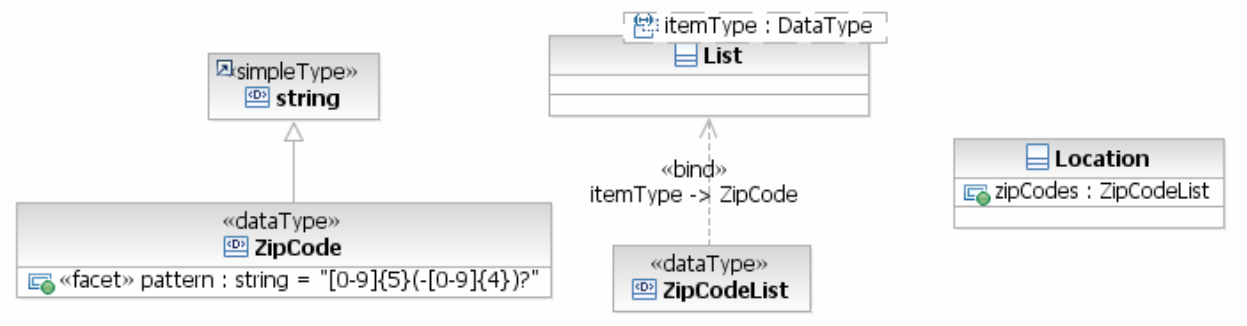
Constraints

- No other stereotypes from this profile may be applied to this DataType.

Semantics

Notation and Use

In the following diagram, a ZipCode type is defined with a pattern facet that restricts values to valid zip code strings in the United States. The ZipCodeList represents a list of these ZipCode types, represented by creating a binding to the List template and refining the itemType parameter to use ZipCode. The list type is used to define the zipCodes attribute in a Location class.



The following schema should be created from this model:

```
<xsd:simpleType name="ZipCode">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[0-9]{5}(-[0-9]{4})?"/>
  </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:simpleType name="ZipCodeList">
  <xsd:list itemType="ZipCode"/>
</xsd:simpleType>

<xsd:complexType name="Location">
  <xsd:sequence>
    <xsd:element name="zipCodes" type="ZipCodeList"/>
  </xsd:sequence>
</xsd:complexType>
```

«union» ( )

Stereotype: Union

Base Metaclass

- DataType

Generalizations

- SimpleType

Description

Defines an XSD union simpleType.

Properties

- memberTypes : DataType [1..*]
  The types of members included in this union.

Constraints

- No other stereotypes from this profile may be applied to this DataType.
- The memberTypes must refer to one or more DataTypes that are not stereotyped as a «complexType».

Semantics

Notation and Use

«group» ( )

Stereotype: Group

Base Metaclass

- Class

Generalizations

- XSDComponent

Description

Defines an XSD global named group.

Properties

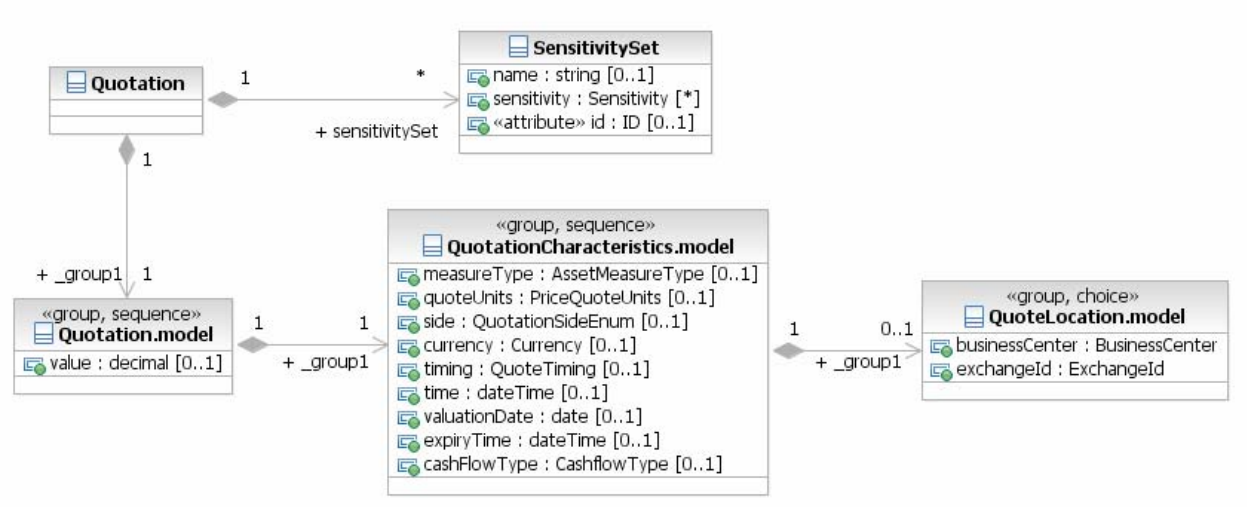
- none

Constraints

-

Semantics

Notation and Use




```
<xsd:complexType name="Quotation">
  <xsd:sequence>
    <xsd:group ref="Quotation.model" />
    <xsd:element name="sensitivitySet" type="SensitivitySet"
                 minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:group name="Quotation.model">
  <xsd:sequence>
    <xsd:element name="value" type="xsd:decimal" minOccurs="0"/>
    <xsd:group ref="QuotationCharacteristics.model"/>
  </xsd:sequence>
</xsd:group>
```

«attributeGroup» ( )

Stereotype: AttributeGroup

Base Metaclass

- Class

Generalizations

- XSDComponent

Description

Properties

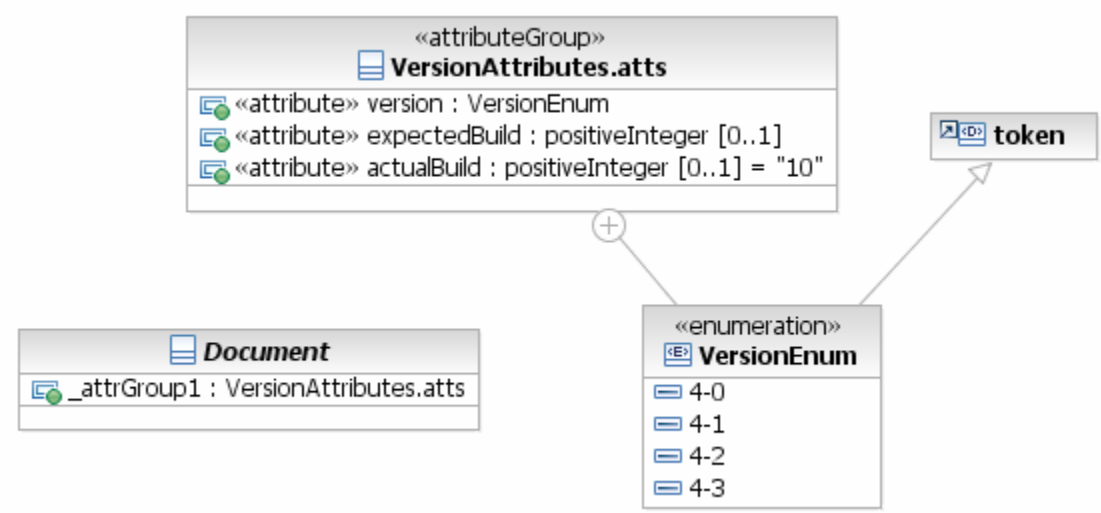
- none

Constraints

-

Semantics

Notation and Use




```
<xsd:complexType abstract="true" name="Document">
  <xsd:attributeGroup ref="VersionAttributes.atts"/>
</xsd:complexType>

<xsd:attributeGroup name="VersionAttributes.atts">
  <xsd:attribute name="version" use="required">
    <xsd:simpleType>
      <xsd:restriction base="xsd:token">
        <xsd:enumeration value="4-0"/>
        <xsd:enumeration value="4-1"/>
        <xsd:enumeration value="4-2"/>
        <xsd:enumeration value="4-3"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="expectedBuild" type="xsd:positiveInteger"/>
  <xsd:attribute name="actualBuild" type="xsd:positiveInteger" fixed="10"/>
</xsd:attributeGroup>
```

«sequence» ( )

Stereotype: Sequence

Base Metaclass

- Class

Generalizations

-

Description

An <xsd:sequence> model group must be used for the containing XSD complexType or group.

Properties

- none

Constraints

- The UML class must be mapped as an XSD complexType or group.
- Only one of «sequence», «choice», or «all» may be applied to a single class.

Semantics

Notation and Use

«choice» ( )

Stereotype: Choice

Base Metaclass

- Class

Generalizations

-

Description

An <xsd:choice> model group must be used for the containing XSD complexType or group.

Properties

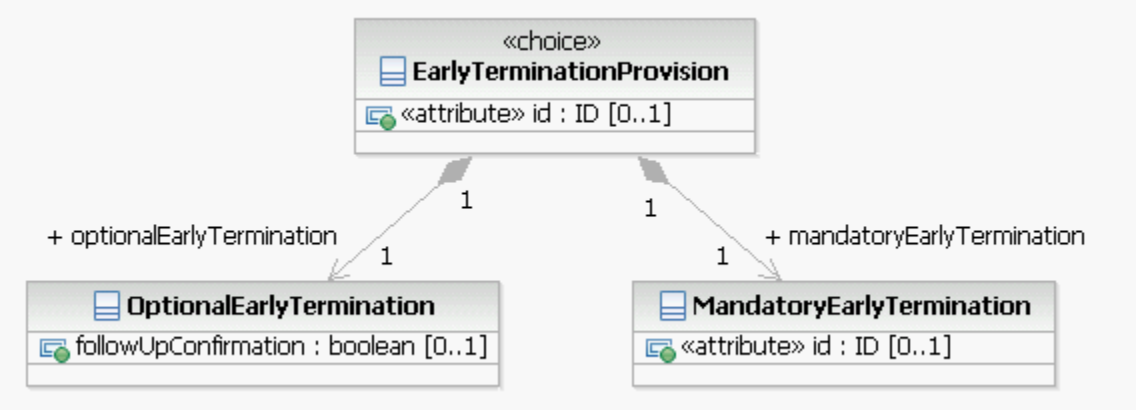
- none

Constraints

- The UML class must be mapped as an XSD complexType or group.
- Only one of «sequence», «choice», or «all» may be applied to a single class.

Semantics

Notation and Use




```
<xsd:complexType name="EarlyTerminationProvision">
  <xsd:choice>
    <xsd:element name="mandatoryEarlyTermination"
                 type="MandatoryEarlyTermination">
    <xsd:element name="optionalEarlyTermination"
                 type="OptionalEarlyTermination">
    </xsd:element>
  </xsd:choice>
  <xsd:attribute name="id" type="xsd:ID"/>
</xsd:complexType>
```

«all» ( )

Stereotype: All

Base Metaclass

- Class

Generalizations

-

Description

An <xsd:all> model group must be used for the containing XSD complexType or group.

Properties

- none

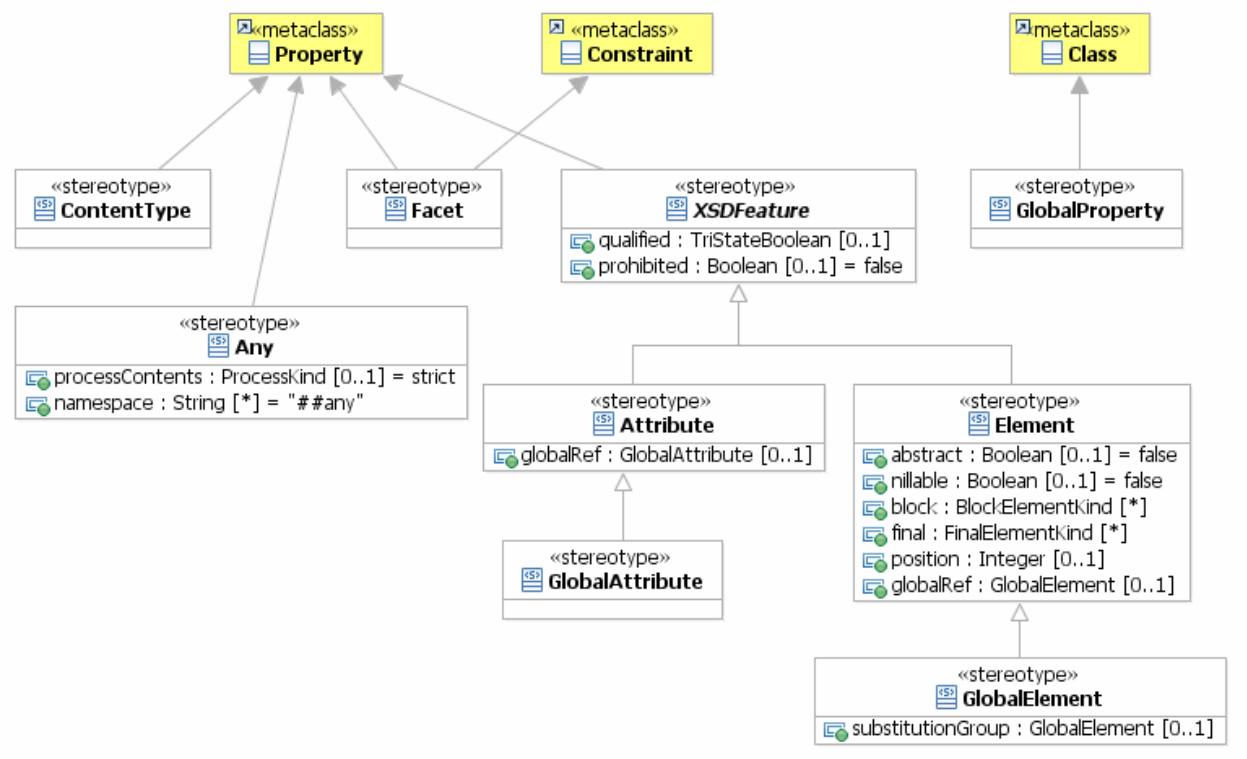
Constraints

- The UML class must be mapped as an XSD complexType or group.
- Only one of «sequence», «choice», or «all» may be applied to a single class.

Semantics

Notation and Use

Property



«XSDFeature»

Base Metaclass

- Property

Generalizations

- XSDComponent

Description

Abstract stereotype for common properties shared by XML Schema elements and attributes.

Properties

- qualified : TriStateBoolean [0..1]
  Indicates whether this feature must be namespace qualified in the schema. Overrides default package setting.
- prohibited : Boolean [0..1]
  This feature is prohibited in the schema type. Used to model XML Schema complexType restrictions.

«element» ( )

Stereotype: Element

Base Metaclass

- Property

Generalizations

- XSDFeature

Description

This property must be mapped to an XSD element in the containing complexType or group.

Properties

- abstract : Boolean [0..1] = false
- nillable : Boolean [0..1] = false
- block : BlockElementKind [0..*]
- final : FinalElementKind [0..*]
- position : Integer [0..1]
  Position of this element within a sequence group. Only required when this property is a navigable end owned by a UML Association.
- globalRef : GlobalElement [0..1]
  Reference to a shared, global element definition. The value must be a Property that has the «gloalElement» stereotype applied.

Constraints

- The only other stereotype that may be applied is «any».

Semantics

Recommend use of navigable Property owned by a Class, instead of owned by Association, because the latter does not allow ordering the content of XSD sequence groups.

When a default value is assigned to the UML property, it is reflected as the XSD element default:

```
<xsd:element name="actualBuild" type="xsd:positiveInteger"
             default="10" minOccurs="0"/>
```

When a default value is assigned and the UML property has `readOnly=true`, then the XSD element has a fixed value:

```
<xsd:element name="actualBuild" type="xsd:positiveInteger"
             fixed="10" minOccurs="0"/>
```

Notation and Use

«globalElement» ( )

Stereotype: GlobalElement

Base Metaclass

- Property

Generalizations

- Element

Description

This property is mapped to a globally defined element in an XML schema.

Properties

- substitutionGroup : GlobalElement [0..1]
  Reference to a shared, global element definition. The value must be a Property that has the «gloalElement» stereotype applied.

Constraints

- No other stereotypes may be applied.
- Must be owned by a class that has the «gloal» stereotype applied.

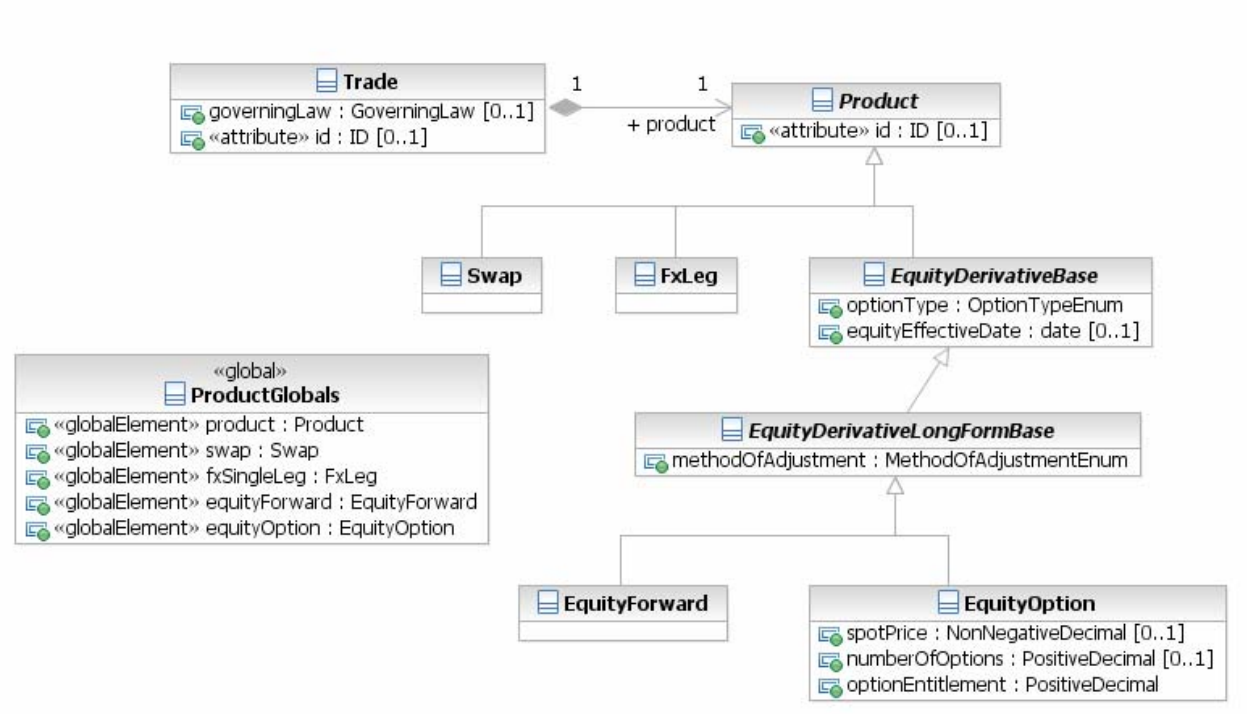
Semantics

Notation and Use

This example is a subset of the FpML schema standard. In the `Trade` class, the `product` attribute has the <<element>> stereotype applied and the `globalRef` value of that stereotype is assigned a reference to the <<globalElement>> stereotype application on the `product` attribute contained in the `ProductGlobals` class.

A UML property may not be declared globally within a package, so the XSD global element and global attribute definitions must be owned by a class. In our profile, this class must be stereotyped with <<globalProperty>> and it contains one or more attributes that have been assigned the <<globalElement>> or <<globalAttribute>> stereotype.

To complete this example, we need to represent the global element substitution group references that are shown in the FpML snippet. The <<globalElement>> stereotype includes a `substitutionGroup` attribute that may refer to another GlobalElement stereotype application. The `swap`, `fxSingleLeg`, `equityForward`, and `equityOption` attributes have this attribute assigned with a reference to the stereotype application on `product` in that same `ProductGlobals` class.




```
<xsd:element name="product" type="Product" abstract="true" />
<xsd:element name="swap" type="Swap" substitutionGroup="product" />
<xsd:element name="fxSingleLeg" type="FxLeg" substitutionGroup="product" />
<xsd:element name="equityForward" type="EquityForward"
                substitutionGroup="product" />
<xsd:element name="equityOption" type="EquityOption"
                substitutionGroup="product" />

<xsd:complexType name="Trade">
  <xsd:sequence>
    <xsd:element name="governingLaw" type="GoverningLaw" minOccurs="0" />
    <xsd:element ref="product" />
  <xsd:sequence>
  <xsd:attribute name="id" type="xsd:ID" />
</xsd:complexType>
```

«attribute» ( )

Stereotype: Attribute

Base Metaclass

- Property

Generalizations

- XSDFeature

Description

The property of the containing classifier is mapped to an XSD attribute.

Properties

- globalRef : GlobalAttribute [0..1]

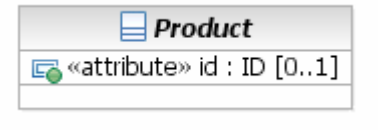
Reference to a shared, global attribute definition. The value must be a Property that has the «gloalAttribute» stereotype applied

Constraints

-

Semantics

Notation and Use




```
<xsd:complexType abstract="true" name="Product">
  <xsd:attribute name="id" type="xsd:ID"/>
</xsd:complexType>
```

«globalAttribute» ( )

Stereotype: GlobalAttribute

Base Metaclass

- Property

Generalizations

- Attribute

Description

This property is mapped to a globally defined attribute in an XML schema.

Properties

- none

Constraints

- No other stereotypes may be applied.
- Must be owned by a class that has the «gloal» stereotype applied.

Semantics

Notation and Use

This example (not from FpML) illustrates the definition of two global attributes. In the `Order` class, the `identifier` attribute has the <<attribute>> stereotype applied and the `globalRef` value of that stereotype is assigned a reference to the <<globalAttribute>> stereotype application on the `identifier` attribute contained in the `Fields` class.


```
<xsd:attribute name="identifier" type="xsd:string" />
<xsd:attribute name="reference" type="xsd:anyURI" />

<xsd:complexType name="Order">
  <xsd:attribute ref="identifier" use="required" />
</xsd:complexType>
```

«global» ( )

Stereotype: GlobalProperty

Base Metaclass

- Class

Generalizations

-

Description

Use of this stereotype on a class that owns properties stereotyped as either «globalElement» or «globalAttribute».

Properties

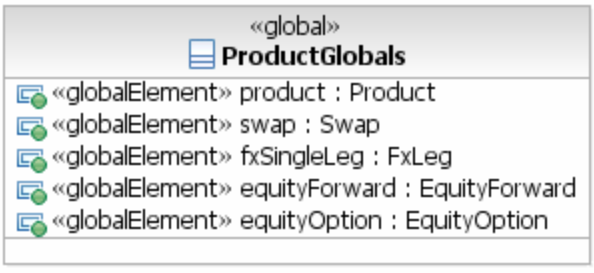
-

Constraints

- All owned attributes must have either «globalElement» or «globalAttribute» applied.

Semantics

Notation and Use




```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:element name="product" type="Product" abstract="true" />
  <xsd:element name="swap" type="Swap" substitutionGroup="product" />
```

```
<xsd:element name="fxSingleLeg" type="FxLeg" substitutionGroup="product" />
<xsd:element name="equityForward" type="EquityForward"
                substitutionGroup="product" />
<xsd:element name="equityOption" type="EquityOption"
                substitutionGroup="product" />
```

«any» ( )

Stereotype: Any

Base Metaclass

- Property

Generalizations

-

Description

Defines wildcard content in an XSD complexType.

Properties

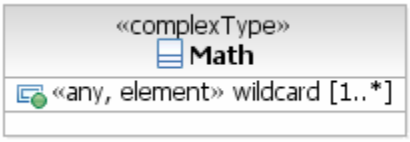
- processContents : ProcessKind [0..1] = strict
- namespace : String [0..*] = ##any

Constraints

- Must be applied in combination with either «element» or «attribute».

Semantics

Notation and Use



In this FpML example, the «complexType» stereotype has attribute `mixed=true`. The «any» stereotype has `processContents` set to “skip” and `namespace` set to “##any”. The attribute name, “wildcard” in this example, is irrelevant to XSD and may be set to any name meaningful to the modeler.

An XSD anyAttribute wildcard is modeled similarly, except that «attribute» is applied instead of «element».

```
<xsd:complexType name="Math" mixed="true">
  <xsd:annotation>
    <xsd:documentation>A type defining a mathematical expression.
      </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:any namespace="##any" processContents="skip" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

«content» ( )

Stereotype: ContentType

Base Metaclass

- Property

Generalizations

-

Description

Defines the content type for an XSD complexType with simpleContent.

Properties

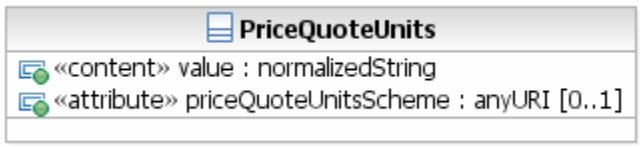
- none

Constraints

- Property must be owned by a Class or DataType mapped to XSD complexType.
- No other stereotypes from this profile may be applied to this Property.
- No more than one Property within the same classifier may have this stereotype applied.
- The type of this property must refer to a DataType mapped to XSD simpleType.

Semantics

Notation and Use




```
<xsd:complexType name="PriceQuoteUnits">
  <xsd:simpleContent>
    <xsd:extension base="xsd:normalizedString">
      <xsd:attribute name="priceQuoteUnitsScheme" type="xsd:anyURI"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

«facet» ( )

Stereotype: Facet

Base Metaclass

- Property
- Constraint

Generalizations

- XSDComponent

Description

Defines a restriction facet on an XSD simpleType.

Properties

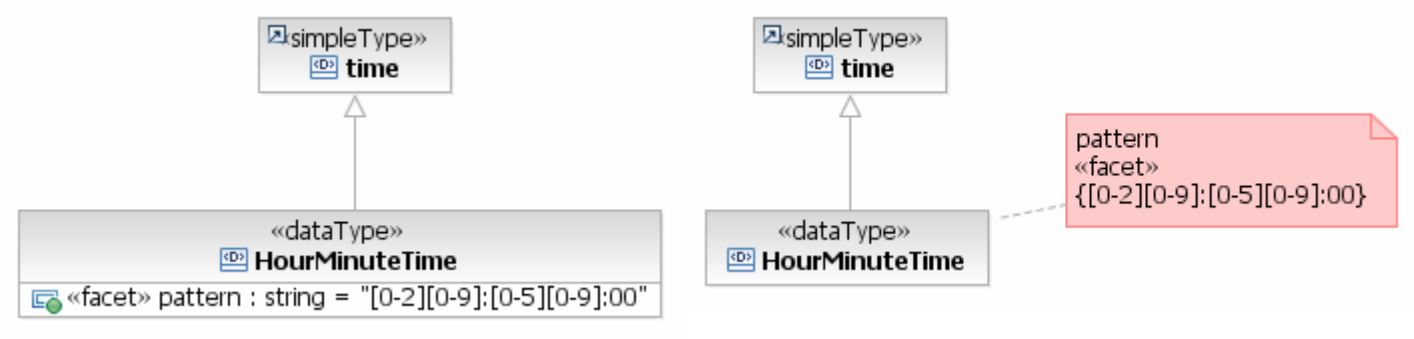
- none

Constraints

- May be applied to a Property or a Constraint owned by a DataType that is mapped to XSD simpleType.

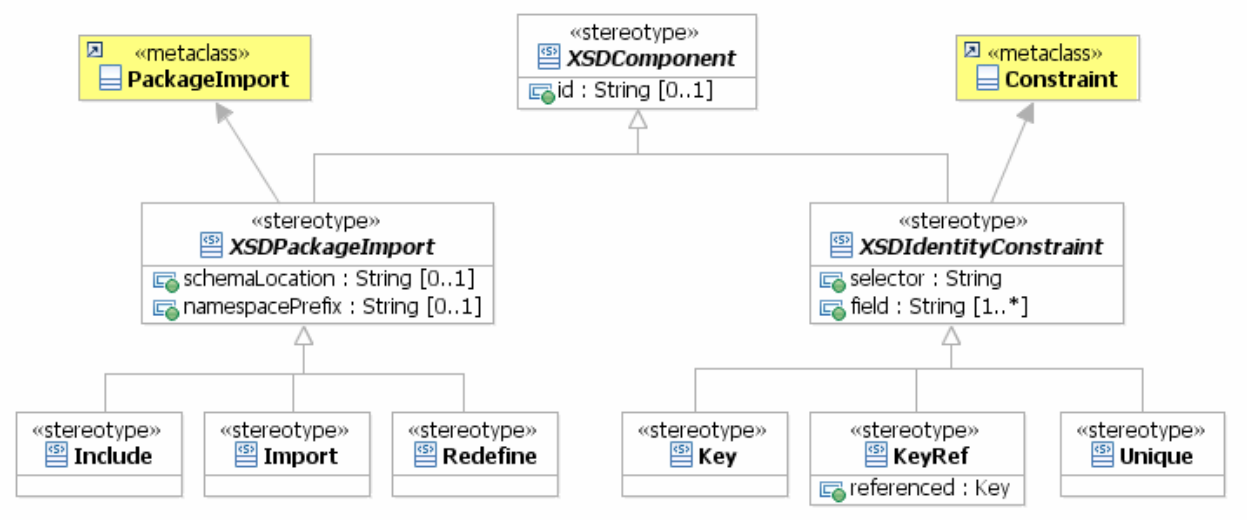
Semantics

Notation and Use




```
<xsd:simpleType name="HourMinuteTime">
  <xsd:restriction base="xsd:time">
    <xsd:pattern value="[0-2][0-9]:[0-5][0-9]:00"/>
  </xsd:restriction>
</xsd:simpleType>
```

PackageImport and Constraint



«stereotype» ( )

Stereotype:

Base Metaclass

-

Generalizations

-

Description

Properties

-

Constraints

-

Semantics

Notation and Use

Enumerations

The following enumeration types are defined and used within this profile for stereotype property types.

TriStateBoolean

- Unspecified
- true
- false

ProcessKind

- strict
- lax
- skip

FormKind

- Unspecified
- qualified
- unqualified

UseKind

- Unspecified
- optional
- required
- prohibited
- default
- fixed

BlockSchemaKind

- ##all
- extension
- restriction
- substitution

FinalSchemaKind

- ##all
- extension
- restriction
- list
- union

BlockComplexTypeKind

- ##all
- extension
- restriction

FinalComplexTypeKind

- ##all
- extension
- restriction

FinalSimpleTypeKind

- ##all
- restriction
- list
- union

BlockElementKind

- ##all
- extension
- restriction
- substitution

FinalElementKind

- ##all
- extension
- restriction

Gaps in XML Schema Mapping

The following are known gaps in the complete mapping from UML to XML Schema:

- XSD Notation
-

References

[1] UML 2.1.1

[2] W3C XML Schema