UML Profile for NIEM

15 February 2011
Version 1.0

URI: http://reference.niem.gov/niem/specification/uml-profile/1.0/

NIEM Technical Architecture Committee (NTAC)

Change History

No.	Date	Reference: All, Page, Table, Figure, Paragraph	A = Add. M = Mod. D = Del.	Revised By	Change Description
1.0	01/15/2011	All	A	NTAC	Initial

Contents

1 Introduction

In the National Information Exchange Model (NIEM) Information Exchange Package Documentation (IEPD) lifecycle, the NIEM modeling process proceeds in two phases: the "Analyze Requirements" phase and the "Map and Model" phase. In the "Analyze Requirements" phase, the user builds a model of their domain. In the "Map and Model" phase, the user maps their model to NIEM.

Often, the user represents the model of their domain in UML. However, before this specification, the semantics of such a model in NIEM had not been defined. Absent that definition, interpretation of such a model would vary; a user could not ensure that the model they produced in the "Analyze Requirements" phase would be interpreted as expected in the "Map and Model" phase.

Further, if the user represented the model of their domain in UML, the user could not convey NIEM concepts without an analogous UML representation. Thus, if the user wished to include NIEM concepts such as type augmentation or element substitution in their model, the user would have to note that fact separately from the UML representation of the model.

1.1 Scope

This specification addresses both issues. It describes the normative UML Profile for NIEM, which consists of a subset of UML 2.3 and a set of extensions to UML 2.3. The subset identifies those NIEM concepts for which an analogous representation exists in UML; use of this subset ensures that a model produced by one user will be interpreted as expected by another user. For those NIEM concepts without an analogous representation in UML, the set of extensions to UML defines such a representation; use of these extensions permit a user to represent NIEM concepts within the UML representation of their model.

A developer should employ the UML Profile for NIEM to construct a UML representation of a NIEM model. To ensure the correct interpretation of a model, tools that produce a UML representation of a NIEM model should employ the profile; tools that consume UML representations of NIEM models should, if the model employs the profile, interpret the model as described in this specification.

While nothing in this specification precludes the application of the profile in other contexts, its focus is the development of the NIEM core model and NIEM domain models. Its use in exchange models should be similar, but has not been tested, and thus may not address their unique requirements. Exchange model developers are encouraged to provide feedback on their

use of the profile so that future revisions of this specification and the profile may address their needs.

1.2 Relationship to NIEM Naming and Design Rules (NDR)

The NIEM data model is implemented in XML Schema. The NIEM Naming and Design Rules (NDR) provides an overview of the NIEM conceptual model, maps from the NIEM conceptual model to XML Schema, and defines a set of rules for XML schemas which enforces the NIEM conceptual model. XML Schemas that follow those rules are referred to as NIEM-conformant schemas.

Like the NIEM NDR maps from the NIEM conceptual model to XML Schema, this document maps from the NIEM conceptual model to UML. However, this document does not define a similar set of rules for defining NIEM conformance in the resulting UML. Instead, it defines a mapping from UML to XML Schema; the XML Schema produced by mapping a UML representation of a NIEM model to XML Schema must adhere to the set of rules defined in NIEM NDR, and thus must be NIEM-conformant.

1.3 Terminology

The user should interpret the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" as described in **[RFC2119]**.

1.4 Normative References

- **[RFC2119]**: Key words for use in RFCs to Indicate Requirement Levels, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997
- **[XMI]**: MOF 2.0/XMI Mapping, Version 2.1.1 (http://www.omg.org/cgi-bin/doc?formal/2007-12-01)
- **[NIEM-NDR]**: National Information Exchange Model Naming and Design Rules, Version 1.3 (http://reference.niem.gov/niem/specification/naming-and-design-rules/1.3/)
- **[UMLSuperstructure]**: OMG Unified Modeling Language, Superstructure, Version 2.3. (http://www.omg.org/spec/UML/2.3/Superstructure/PDF/)
- **[UMLInfrastructure]**: OMG Unified Modeling Language, Infrastructure, Version 2.3. (http://www.omg.org/spec/UML/2.3/Infrastructure/PDF/)
- **[XMLSchemaDatatypes]** - XML Schema Part 2: Datatypes Second Edition, W3C Recommendation (http://www.w3.org/TR/xmlschema-2/)
- **[XMLSchemaStructures]** - XML Schema Part 1: Structures Second Edition, W3C Recommendation (http://www.w3.org/TR/xmlschema-1/)

2 UML Subset and Extension

The UML Profile for NIEM extends the standard UML package Classes defined in Section 7 of **[UMLSuperstructure]**. From the Classes package, the profile uses the following metaclasses:

Table 1: UML Modeling Concepts	
Metaclass	**Section for Metaclass in [UMLSuperstructure]**
Class	7.3.7
DataType	7.3.11
Dependency	7.3.12
Enumeration	7.3.16
EnumerationLiteral	7.3.17
Generalization	7.3.20
Package	7.3.37
Property	7.3.44

The UML Profile for NIEM extends these metaclasses to convey NIEM concepts without an analogous representation in a UML model. The following table lists each stereotype and the base metaclass it extends:

Table 2: Stereotypes	
Stereotype	**Base Metaclass**
NIEMNamespace	Package
NIEMSimpleType	DataType
NIEMCSCType	DataType
NIEMCodeSimpleType	Enumeration
NIEMCodeType	Enumeration
NIEMCodeValue	EnumerationLiteral
NIEMCCCType	Class
NIEMObjectType	Class
NIEMAssociationType	Class
NIEMMetadataType	Class

NIEMMetadataApplication	Dependency
NIEMAugmentationType	Class
NIEMAdapterType	Class
NIEMExternalProperty	Property
NIEMProperty	Property
NIEMTopLevel	Class

Each section of this specification has the following structure.

First, the specification identifies the stereotype, its generalization, and its analogous NIEM concept; the specification then broadly describes the implementation of the analogous NIEM concept in XML Schema. For example:

> The NIEMNamespace stereotype is a specialization of Package. It represents a NIEM namespace, which is implemented in XML Schema as an XML schema document.

Second, the specification maps from those attributes and associations of the generalization that have an analogous NIEM concept to their analogous NIEM concepts; if no attribute or association of the generalization has an analogous NIEM concept, the specification notes this. For example:

> `packagedElement`: PackageableElement [0..*]
>
> > The reference to the components in the NIEM namespace, implemented in XML Schema as defined in this specification.

Third, the specification defines the attributes and associations of the stereotype: the name of the attribute or association, the type of the attribute or association, the multiplicity of the attribute or association, the description of the attribute or association. The description of the attribute or association includes the analogous NIEM concept and the implementation of the analogous NIEM concept in XML Schema. For example:

> `definition`: String [0..1]
>
> > The data definition for the NIEM namespace, implemented in XML Schema as the content of the first element `xsd:documentation` on the document element `xsd:schema`. See Section 7.2.1 of **[NIEM-NDR]**, Rule 7-9.

Fourth, the specification defines any constraints that may apply to the stereotype. For example:

If the NIEMSimpleType is not the specific Classifier in a Generalization and the attribute `base` is empty, the base type definition for the simple type definition that is the implementation of the NIEM type shall be xsd:token.

Note that while the specification derives these constraints from **[NIEM-NDR]**, **[XMLSchemaStructures]**, and **[XMLSchemaDatatypes]** and adherence to these constraints is necessary to produce a NIEM-conformant schema, these constraints do not reflect every rule described in those specifications, and thus adherence to these constraints is not sufficient to produce a NIEM-conformant schema. For example, these constraints do not specify that the attribute `name` of Class adhere to the rules for the name of a NIEM type or the name property of a complex or simple type definition, although adherence to those rules is necessary to produce a NIEM-conformant schema. Instead, the intent of these constraints is to provide additional guidance to the user.

Finally, the specification provides an illustration of the use of the stereotype and the implementation of that illustration in XML Schema. In these examples, the prefix xsd refers to the namespace `http://www.w3.org/2001/XMLSchema`, the prefix s refers to the namespace specified by the URI `http://niem.gov/niem/structures/2.0`, the prefix i refers to the namespace specified by the URI `http://niem.gov/niem/appinfo/2.0`.

2.1 NIEMNamespace

Description

The NIEMNamespace stereotype is a specialization of Package. It represents a NIEM namespace, which is implemented in XML Schema as an XML schema document.

Attributes and Associations

NIEMNamespace inherits the following association from Package; it shall map to an analogous NIEM concept and XML Schema implementation as described:

`packagedElement` PackageableElement [0..*]

The reference to the components in the NIEM namespace, implemented in XML Schema as defined in this specification.

NIEMNamespace defines the following attributes, which shall map to an analogous NIEM concept and XML Schema implementation as described:

`definition`: String [0..1]

The data definition for the NIEM namespace, implemented in XML Schema as the content of the first element `xsd:documentation` on the document element `xsd:schema`. See Section 7.2.1 of **[NIEM-NDR]**, Rule 7-9.

`isConformant` Boolean [0..1]

Indicates that the NIEM namespace is NIEM-conformant, implemented in XML Schema as the content of the application information `i:ConformantIndicator` of the document element `xsd:schema`. Default is true. See Section 7.1 of **[NIEM-NDR]**, Rule 7-1.

`namespace`: String [0..1]

The target namespace of the NIEM namespace, implemented in XML Schema as the value of the attribute `targetNamespace` on the document element `xsd:schema`. See Section 6.2 of **[NIEM-NDR]**, Rule 6-35 and Rule 6-36.

`version`: String [0..1]

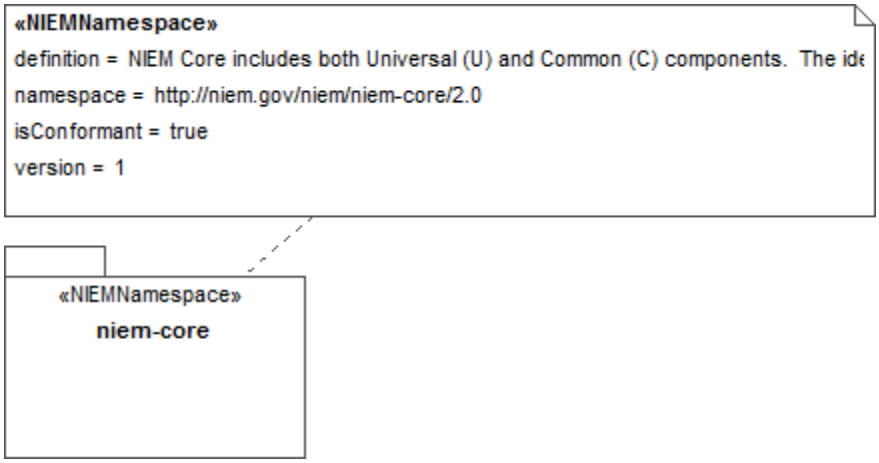
The version of the NIEM namespace, implemented in XML Schema as the value of the attribute `version` on the document element `xsd:schema`. Default is 1. See Section 6.2 of **[NIEM-NDR]**, Rule 6-37 and Rule 6-38.

Constraints

NIEMNamespace does not adhere to any additional constraints.

Examples

The following diagram illustrates the use of the stereotype:



The above diagram is implemented in XML Schema as follows:

```
<xsd:schema targetNamespace="http://niem.gov/niem/niem-core/2.0" version="1">
    <xsd:annotation>
        <xsd:documentation>
            NIEM Core includes both Universal (U) and Common (C) components.
            The identities for U and C components in Core are maintained with
            metadata.
```

```
        </xsd:documentation>
        <xsd:appinfo>
            <i:ConformantIndicator>true</i:ConformantIndicator>
        </xsd:appinfo>
    </xsd:annotation>
</xsd:annotation>
```

2.2 **NIEMSimpleType**

Description

The NIEMSimpleType stereotype is a specialization of DataType. It represents a NIEM type which is implemented in XML Schema as a simple type definition. See Section 7.3 of **[NIEM-NDR]** and Section 3.14 of **[XMLSchemaStructures]**.

Attributes and Associations

NIEMSimpleType inherits the following attribute from DataType; it shall map to an analogous NIEM concept and XML Schema implementation as described:

`name`: String [0..1]

> The name of the NIEM type, implemented in XML Schema as the value of the attribute `name` on the element `xsd:simpleType`.

NIEMSimpleType defines the following attributes, which shall map to an analogous NIEM concept and XML Schema implementation as described:

`base`: String [0..1]

> The name of the base type definition of the NIEM type, implemented in XML Schema as the value of the attribute `base` on the element `xsd:restriction`, the immediate child of the element `xsd:simpleType`.

`definition`: String [0..1]

> The data definition for the NIEM type, implemented in XML Schema as the content of the first element `xsd:documentation` on the element `xsd:simpleType`. See Section 7.2.1 of **[NIEM-NDR]**, Rule 7-5.

`fractionDigits`: Integer [0..1]

> A restriction on the value space of the NIEM type, implemented in XML Schema as the value of the attribute `value` on the element `xsd:fractionDigits`, the child of the element `xsd:restriction` which is the immediate child of the element `xsd:simpleType`. See Section 4.3.12 of **[XMLSchemaDatatypes]**.

`length`: Integer [0..1]

A restriction on the value space of the NIEM type, implemented in XML Schema as the value of the attribute `value` on the element `xsd:length`, the child of the element `xsd:restriction` which is the immediate child of the element `xsd:simpleType`. See Section 4.3.1 of **[XMLSchemaDatatypes]**.

`maxExclusive`: String [0..1]

A restriction on the value space of the NIEM type, implemented in XML Schema as the value of the attribute `value` on the element `xsd:maxExclusive`, the child of the element `xsd:restriction` which is the immediate child of the element `xsd:simpleType`. See Section 4.3.8 of **[XMLSchemaDatatypes]**.

`maxInclusive`: String [0..1]

A restriction on the value space of the NIEM type, implemented in XML Schema as the value of the attribute `value` on the element `xsd:maxInclusive`, the child of the element `xsd:restriction` which is the immediate child of the element `xsd:simpleType`. See Section 4.3.7 of **[XMLSchemaDatatypes]**.

`maxLength`: Integer [0..1]

A restriction on the value space of the NIEM type, implemented in XML Schema as the value of the attribute `value` on the element `xsd:maxLength`, the child of the element `xsd:restriction` which is the immediate child of the element `xsd:simpleType`. See Section 4.3.3 of **[XMLSchemaDatatypes]**.

`minExclusive`: String [0..1]

A restriction on the value space of the NIEM type, implemented in XML Schema as the value of the attribute `value` on the element `xsd:minExclusive`, the child of the element `xsd:restriction` which is the immediate child of the element `xsd:simpleType`. See Section 4.3.9 of **[XMLSchemaDatatypes]**.

`minInclusive`: String [0..1]

A restriction on the value space of the NIEM type, implemented in XML Schema as the value of the attribute `value` on the element `xsd:minInclusive`, the child of the element `xsd:restriction` which is the immediate child of the element `xsd:simpleType`. See Section 4.3.10 of **[XMLSchemaDatatypes]**.

`minLength`: Integer [0..1]

A restriction on the value space of the NIEM type, implemented in XML Schema as the value of the attribute `value` on the element `xsd:minLength`, the child of the element

`xsd:restriction` which is the immediate child of the element `xsd:simpleType`. See Section 4.3.2 of **[XMLSchemaDatatypes]**.

`pattern`: String [0..1]

A restriction on the value space of the NIEM type, implemented in XML Schema as the value of the attribute `value` on the element `xsd:pattern`, the child of the element `xsd:restriction` which is the immediate child of the element `xsd:simpleType`. See Section 4.3.4 of **[XMLSchemaDatatypes]**.

`totalDigits`: Integer [0..1]

A restriction on the value space of the NIEM type, implemented in XML Schema as the value of the attribute `value` on the element `xsd:totalDigits`, the child of the element `xsd:restriction` which is the immediate child of the element `xsd:simpleType`. See Section 4.3.11 of **[XMLSchemaDatatypes]**.

`whiteSpace`: String [0..1]

A restriction on the value space of the NIEM type, implemented in XML Schema as the value of the attribute `value` on the element `xsd:whiteSpace`, the child of the element `xsd:restriction` which is the immediate child of the element `xsd:simpleType`. See Section 4.3.6 of **[XMLSchemaDatatypes]**.

Constraints

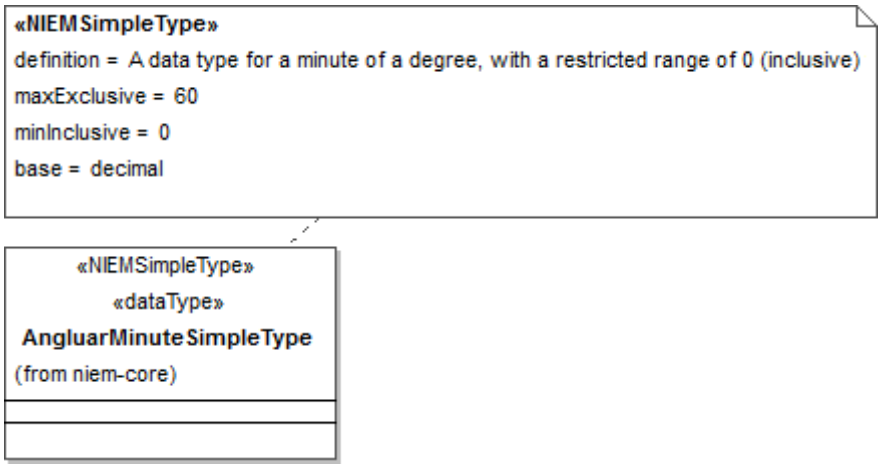
NIEMSimpleType adheres to following constraints:

1. If the NIEMSimpleType is the specific Classifier in a Generalization,
    a. the attribute `base` must be empty;
    b. the general Classifier must be a NIEMSimpleType; and
    c. the base type definition for the simple type definition that is the implementation of the NIEM type shall be the simple type definition that is the implementation of the NIEM type represented by the general Classifier.
2. If the NIEMSimpleType is not the specific Classifier in a Generalization and the attribute `base` is empty, the base type definition for the simple type definition that is the implementation of the NIEM type shall be xsd:token.
3. If the NIEMSimpleType is not the specific Classifier in a Generalization and the attribute `base` is not empty, the base type definition for the simple type definition that is the implementation of the NIEM type shall be the type defined in **[XMLSchemaDatatypes]** with the same name as the value of the attribute `base`.

For example, if the value of the attribute `base` is decimal, the base type definition shall be xsd:decimal.

Examples

The following diagram illustrates the use of the stereotype:



The above diagram is implemented in XML Schema as follows:

```
<xsd:simpleType name="AngularMinuteSimpleType">
    <xsd:annotation>
        <xsd:documentation>
            A data type for a minute of a degree, with a restricted range of
            0 (inclusive) to 60 (exclusive).
        </xsd:documentation>
        <xsd:appinfo>
            <i:Base i:namespace="http://niem.gov/niem/structures/2.0"
                    i:name="Object"/>
        </xsd:appinfo>
    </xsd:annotation>
    <xsd:restriction base="xsd:decimal">
        <xsd:maxExclusive value="60"/>
        <xsd:minInclusive value="0"/>
    </xsd:restriction>
</xsd:simpleType>
```

2.3 NIEMCSCType

Description

The NIEMCSCType stereotype is a specialization of DataType. It represents one of these NIEM concepts:

1. A NIEM type which is implemented in XML Schema by a complex type definition with simple content. See Section 7.4 of **[NIEM-NDR]** and Section 3.4 of **[XMLSchemaStructures]**.

2. A NIEM type which implemented in XML Schema by a simple type definition and a complex type definition with simple content, the base type definition of which is the simple type definition. Sections 7.3 and 7.4 of **[NIEM-NDR]** and Sections 3.4 and 3.14 of **[XMLSchemaStructures]**.

   This is a common pattern in NIEM, and the specification incorporates it into this stereotype as a convenience to the user.

The context of the NIEMCSCType determines the NIEM concept and thus the XML Schema representation.

The acronym CSC in the stereotype name derives from the XML Schema implementation of the NIEM concept, a complex type definition with simple content.

Attributes and Associations

NIEMCSCType inherits the following attribute from DataType; it shall map to an analogous NIEM concept and XML Schema implementation as described:

`name`: String [0..1]

> The name of the NIEM type, implemented in XML Schema as the value of the attribute `name` on the element `xsd:complexType`.

NIEMCSCType defines the following attributes, which shall map to an analogous NIEM concept and XML Schema implementation as described:

`definition`: String [0..1]

> The data definition for the NIEM complex type, implemented in XML Schema as the content of the first element `xsd:documentation` on the element `xsd:complexType`; in the context of second NIEM concept above, also the data definition for the simple type definition, represented in XML Schema as the content of the first element `xsd:documentation` on the element `xsd:simpleType`. See Section 7.2.1 of **[NIEM-NDR]**, Rule 7-4 and Rule 7-5.

The following attributes have meaning only in the context of the second NIEM concept above; they shall map to an analogous NIEM concept and XML Schema implementation as described:

`base`: String [0..1]

The name of the base type definition of the simple type definition, implemented in XML Schema as the value of the attribute `base` on the element `xsd:restriction`, the immediate child of the element `xsd:simpleType`.

`fractionDigits`: Integer [0..1]

A restriction on the value space of the simple type definition, implemented in XML Schema as the value of the attribute `value` on the element `xsd:fractionDigits`, the child of the element `xsd:restriction` which is the immediate child of the element `xsd:simpleType`. See Section 4.3.12 of **[XMLSchemaDatatypes]**.

`length`: Integer [0..1]

A restriction on the value space of the simple type definition, implemented in XML Schema as the value of the attribute `value` on the element `xsd:length`, the child of the element `xsd:restriction` which is the immediate child of the element `xsd:simpleType`. See Section 4.3.1 of **[XMLSchemaDatatypes]**.

`maxExclusive`: String [0..1]

A restriction on the value space of the simple type definition, implemented in XML Schema as the value of the attribute `value` on the element `xsd:maxExclusive`, the child of the element `xsd:restriction` which is the immediate child of the element `xsd:simpleType`. See Section 4.3.8 of **[XMLSchemaDatatypes]**.

`maxInclusive`: String [0..1]

A restriction on the value space of the simple type definition, implemented in XML Schema as the value of the attribute `value` on the element `xsd:maxInclusive`, the child of the element `xsd:restriction` which is the immediate child of the element `xsd:simpleType`. See Section 4.3.7 of **[XMLSchemaDatatypes]**.

`maxLength`: Integer [0..1]

A restriction on the value space of the simple type definition, implemented in XML Schema as the value of the attribute `value` on the element `xsd:maxLength`, the child of the element `xsd:restriction` which is the immediate child of the element `xsd:simpleType`. See Section 4.3.3 of **[XMLSchemaDatatypes]**.

`minExclusive`: String [0..1]

A restriction on the value space of the simple type definition, implemented in XML Schema as the value of the attribute `value` on the element `xsd:minExclusive`, the child of the element `xsd:restriction` which is the immediate child of the element `xsd:simpleType`. See Section 4.3.9 of **[XMLSchemaDatatypes]**.

`minInclusive`: String [0..1]

A restriction on the value space of the simple type definition, implemented in XML Schema as the value of the attribute `value` on the element `xsd:minInclusive`, the child of the element `xsd:restriction` which is the immediate child of the element `xsd:simpleType`. See Section 4.3.10 of **[XMLSchemaDatatypes]**.

`minLength`: Integer [0..1]

A restriction on the value space of the simple type definition, implemented in XML Schema as the value of the attribute `value` on the element `xsd:minLength`, the child of the element `xsd:restriction` which is the immediate child of the element `xsd:simpleType`. See Section 4.3.2 of **[XMLSchemaDatatypes]**.

`simple`: String [0..1]

The name of the simple type definition, implemented in XML Schema as the value of the attribute `name` on the element `xsd:simpleType`.

`pattern`: String [0..1]

A restriction on the value space of the simple type definition, implemented in XML Schema as the value of the attribute `value` on the element `xsd:pattern`, the child of the element `xsd:restriction` which is the immediate child of the element `xsd:simpleType`. See Section 4.3.4 of **[XMLSchemaDatatypes]**.

`totalDigits`: Integer [0..1]

A restriction on the value space of the simple type definition, implemented in XML Schema as the value of the attribute `value` on the element `xsd:totalDigits`, the child of the element `xsd:restriction` which is the immediate child of the element `xsd:simpleType`. See Section 4.3.11 of **[XMLSchemaDatatypes]**.

`whiteSpace`: String [0..1]

A restriction on the value space of the simple type definition, implemented in XML Schema as the value of the attribute `value` on the element `xsd:whiteSpace` which is the child of the element `xsd:restriction` which is the immediate child of the element `xsd:simpleType`. See Section 4.3.6 of **[XMLSchemaDatatypes]**.

Constraints

NIEMCSCType adheres to following constraints:

1. If the NIEMCSCType is the specific Classifier in a Generalization,

a. the NIEMCSCType represents the first NIEM concept above;

b. the attributes `base`, `fractionDigits`, `length`, `maxExclusive`, `maxInclusive`, `maxLength`, `minExclusive`, `minInclusive`, `minLength`, `simple`, `pattern`, and `totalDigits` must be empty;

c. the general Classifier must be a NIEMSimpleType or a NIEMCSCType; and

d. the base type definition of the complex type definition that is the implementation of the NIEM type shall be the type definition that is the implementation of the NIEM type represented by the general Classifier.

2. If the NIEMCSCType is not the specific Classifier in a Generalization, the attribute `simple` is empty, and the attribute `base` is empty

   a. the NIEMCSCType represents the first concept above;

   b. the attributes `fractionDigits`, `length`, `maxExclusive`, `maxInclusive`, `maxLength`, `minExclusive`, `minInclusive`, `minLength`, `pattern`, and `totalDigits` must be empty; and

   c. the base type definition for the complex type definition that is the implementation of the NIEM type shall be xsd:token.

3. If the NIEMCSCType is not the specific Classifier in a Generalization, the attribute `simple` is empty, and the attribute `base` is not empty

   a. the NIEMCSCType represents the first concept above;

   b. the attributes `fractionDigits`, `length`, `maxExclusive`, `maxInclusive`, `maxLength`, `minExclusive`, `minInclusive`, `minLength`, `pattern`, and `totalDigits` must be empty; and

   c. the base type definition for the complex type definition that is the implementation of the NIEM type shall be the type defined in **[XMLSchemaDatatypes]** with the same name as the value of the attribute `base`.

      For example, if the value of the attribute `base` is decimal, the base type definition shall be xsd:decimal.

4. If the NIEMCSCType is not the specific Classifier in a Generalization, the attribute `simple` is not empty, and the attribute `base` is empty,

   a. the NIEMCSCType represents the second concept above;

   b. the base type definition for the simple type definition that is the implementation of the NIEM type shall be xsd:token.

5. If the NIEMCSCType is not the specific Classifier in a Generalization, the attribute `simple` is not empty, and the attribute `base` is not empty,

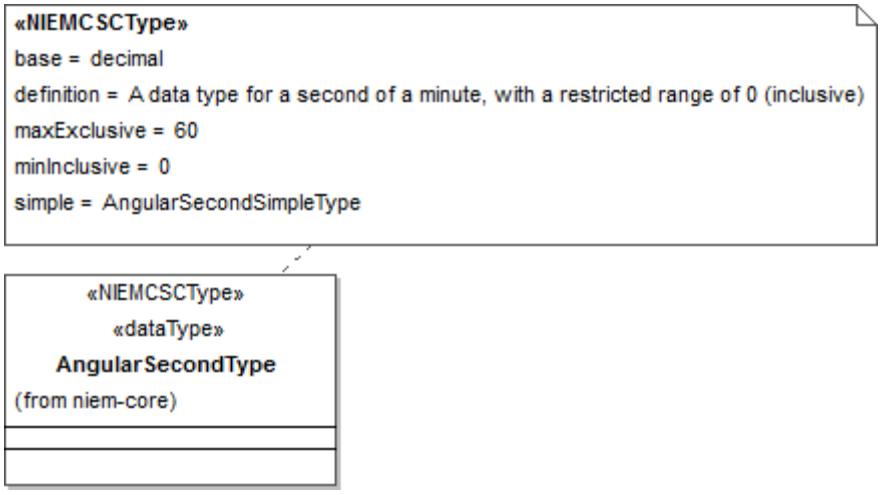
    a. the NIEMCSCType represents the second concept above;

    b. the base type definition for the simple type definition that is the implementation of the NIEM type shall be the type defined in **[XMLSchemaDatatypes]** with the same name as the value of the attribute `base`.

    For example, if the value of the attribute `base` is decimal, the base type definition shall be xsd:decimal.

Examples

The following diagram illustrates the use of the stereotype:



The above diagram is implemented in XML Schema as follows:

```
<xsd:simpleType name="AngularSecondSimpleType">
    <xsd:annotation>
        <xsd:documentation>
            A data type for a second of a minute, with a restricted range of
            0 (inclusive) to 60 (exclusive).
        </xsd:documentation>
        <xsd:appinfo>
            <i:Base i:namespace="http://niem.gov/niem/structures/2.0"
                    i:name="Object"/>
        </xsd:appinfo>
    </xsd:annotation>
    <xsd:restriction base="xsd:decimal">
        <xsd:minInclusive value="0"/>
        <xsd:maxExclusive value="60"/>
    </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="AngularSecondType">
    <xsd:annotation>
```

```
        <xsd:documentation>
            A data type for a second of a minute, with a restricted range of
            0 (inclusive) to 60 (exclusive).
        </xsd:documentation>
        <xsd:appinfo>
            <i:Base i:namespace="http://niem.gov/niem/structures/2.0"
                    i:name="Object"/>
        </xsd:appinfo>
    </xsd:annotation>
    <xsd:simpleContent>
        <xsd:extension base="nc:AngularSecondSimpleType">
            <xsd:attributeGroup ref="s:SimpleObjectAttributeGroup"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
```

2.4 **NIEMCodeSimpleType**

Description

The NIEMCodeSimpleType stereotype is a specialization of Enumeration. It represents a NIEM code type which is implemented in XML Schema as a simple type definition. See Section 7.3 and Section 9.12.3 of **[NIEM-NDR]** and Section 3.14 of **[XMLSchemaStructures]**.

Attributes and Associations

NIEMCodeSimpleType inherits the following attribute from Enumeration; it shall map to an analogous NIEM concept and XML Schema implementation as described:

`name`: String [0..1]

> The name of the NIEM code type, implemented in XML Schema as the value of the attribute `name` on the element `xsd:simpleType`.

NIEMCodeSimpleType defines the following attributes, which shall map to an analogous NIEM concept and XML Schema implementation as described:

`base`: String [0..1]

> The name of the base type definition of the NIEM code type, implemented in XML Schema as the value of the attribute `base` on the element `xsd:restriction`, the immediate child of the element `xsd:simpleType`.

`definition`: String [0..1]

> The data definition for the NIEM code type, implemented in XML Schema as the content of the first element `xsd:documentation` on the element `xsd:simpleType`. See Section 7.2.1 of **[NIEM-NDR]**, Rule 7-5.

Constraints

NIEMCodeSimpleType has the following constraints:

1. If the NIEMCodeSimpleType is the specific Classifier in a Generalization,

    a. the attribute `base` must be empty;

    b. the general Classifier must be a NIEMCodeSimpleType; and

    c. the base type definition for the simple type definition that is the implementation of the NIEM type shall be the simple type definition that is the implementation of the NIEM type represented by the general Classifier.

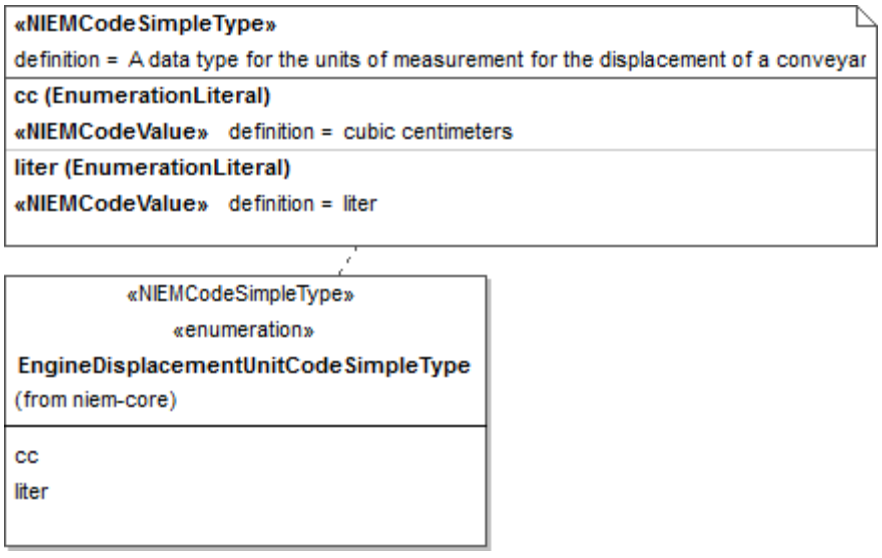
2. If the NIEMCodeSimpleType is not the specific Classifier in a Generalization and the attribute `base` is empty, the base type definition for the simple type definition that is the implementation of the NIEM type shall be xsd:token.

3. If the NIEMCodeSimpleType is not the specific Classifier in a Generalization and the attribute `base` is not empty, the base type definition for the simple type definition that is the implementation of the NIEM type shall be the type defined in **[XMLSchemaDatatypes]** with the same name as the value of the attribute `base`.

    For example, if the value of the attribute `base` is decimal, the base type definition shall be xsd:decimal.

Examples

The following diagram illustrates the use of the stereotype:



The above diagram is implemented in XML Schema as follows:

```
<xsd:simpleType name="EngineDisplacementUnitCodeSimpleType">
    <xsd:annotation>
        <xsd:documentation>
            A data type for the units of measurement for the displacement of
            a conveyance engine.
        </xsd:documentation>
        <xsd:appinfo>
            <i:Base i:namespace="http://niem.gov/niem/structures/2.0"
                    i:name="Object"/>
        </xsd:appinfo>
    </xsd:annotation>
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="cc">
            <xsd:annotation>
                <xsd:documentation>cubic centimeters</xsd:documentation>
            </xsd:annotation>
        </xsd:enumeration>
        <xsd:enumeration value="liter">
            <xsd:annotation>
                <xsd:documentation>liter</xsd:documentation>
            </xsd:annotation>
        </xsd:enumeration>
    </xsd:restriction>
</xsd:simpleType>
```

2.5 **NIEMCodeType**

Description

The NIEMCodeType stereotype is a specialization of Enumeration. It represents one of these NIEM concepts:

1. A NIEM code type which is implemented in XML Schema by a complex type definition with simple content. See Sections 7.3 and 9.12.3 of **[NIEM-NDR]** and Section 3.4 of **[XMLSchemaStructures]**.

2. A NIEM code type implemented in XML Schema by a simple type definition and a complex type definition with simple content, the base type definition of which is the simple type definition. See Sections 7.3, 7.4, and 9.12.3 of **[NIEM-NDR]** and Sections 3.4 and 3.14 of **[XMLSchemaStructures]**.

   This is a common pattern in NIEM, and the specification incorporates it into this stereotype as a convenience to the user.

The context of the NIEMCodeType determines the NIEM concept and thus the XML Schema representation.

Attributes and Associations

NIEMCodeType inherits the following attribute from Enumeration; it shall map to an analogous NIEM concept and XML Schema implementation as described:

`name`: String [0..1]

> The name of the NIEM type, implemented in XML Schema as the value of the attribute `name` on the element `xsd:complexType`.

NIEMCodeType defines the following attributes, which shall map to an analogous NIEM concept and XML Schema implementation as described:

`definition`: String [0..1]

> The data definition for the NIEM complex type, implemented in XML Schema as the content of the first element `xsd:documentation` on the element `xsd:complexType`; in the context of second NIEM concept above, also the data definition for the simple type definition, represented in XML Schema as the content of the first element `xsd:documentation` on the element `xsd:simpleType`. See Section 7.2.1 of **[NIEM-NDR]**, Rule 7-4 and Rule 7-5.

The following attributes have meaning only in the context of the second NIEM concept above; they shall map to an analogous NIEM concept and XML Schema implementation as described:

`base`: String [0..1]

> The name of the base type definition of the simple type definition, implemented in XML Schema as the value of the attribute `base` on the element `xsd:restriction`, the immediate child of the element `xsd:simpleType`.

`simple`: String [0..1]

> The name of the simple type definition, implemented in XML Schema as the value of the attribute `name` on the element `xsd:simpleType`.

Constraints

NIEMCodeType has the following constraints:

1. If the NIEMCodeType is the specific Classifier in a Generalization,
    a. the NIEMCodeType represents the first NIEM concept above;
    b. the attributes `base` and `simple` must be empty;

c. the general Classifier must be a NIEMCodeSimpleType or a NIEMCodeType; and

d. the base type definition of the complex type definition that is the implementation of the NIEM type shall be the type definition that is the implementation of the NIEM type represented by the general Classifier.

2. If the NIEMCodeType is not the specific Classifier in a Generalization, the attribute `simple` is empty, and the attribute `base` is empty

   a. the NIEMCodeType represents the first concept above and

   b. the base type definition for the complex type definition that is the implementation of the NIEM type shall be xsd:token.

3. If the NIEMCodeType is not the specific Classifier in a Generalization, the attribute `simple` is empty, and the attribute `base` is not empty

   a. the NIEMCodeType represents the first concept above;

   b. the base type definition for the complex type definition that is the implementation of the NIEM type shall be the type defined in **[XMLSchemaDatatypes]** with the same name as the value of the attribute `base`.

      For example, if the value of the attribute `base` is decimal, the base type definition shall be xsd:decimal.

4. If the NIEMCodeType is not the specific Classifier in a Generalization, the attribute `simple` is not empty, and the attribute `base` is empty,

   a. the NIEMCodeType represents the second concept above;

   b. the base type definition for the simple type definition that is the implementation of the NIEM type shall be xsd:token.

5. If the NIEMCodeType is not the specific Classifier in a Generalization, the attribute `simple` is not empty, and the attribute `base` is not empty,

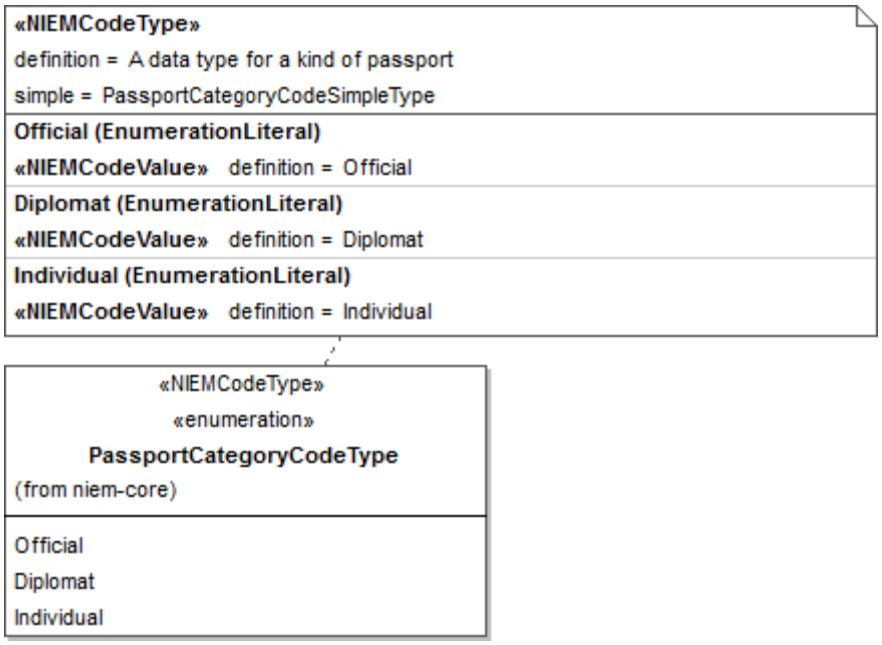
   a. the NIEMCodeType represents the second concept above;

   b. the base type definition for the simple type definition that is the implementation of the NIEM type shall be the type defined in **[XMLSchemaDatatypes]** with the same name as the value of the attribute `base`.

      For example, if the value of the attribute `base` is decimal, the base type definition shall be xsd:decimal.

Examples

The following diagram illustrates the use of the stereotype:



The above diagram is implemented in XML Schema as follows:

```
<xsd:simpleType name="PassportCategoryCodeSimpleType">
    <xsd:annotation>
        <xsd:documentation>
            A data type for a kind of passport.
        </xsd:documentation>
        <xsd:appinfo>
            <i:Base i:namespace="http://niem.gov/niem/structures/2.0"
             i:name="Object"/>
        </xsd:appinfo>
    </xsd:annotation>
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="Official">
            <xsd:annotation>
                <xsd:documentation>Official</xsd:documentation>
            </xsd:annotation>
        </xsd:enumeration>
        <xsd:enumeration value="Diplomat">
            <xsd:annotation>
                <xsd:documentation>Diplomat</xsd:documentation>
            </xsd:annotation>
        </xsd:enumeration>
        <xsd:enumeration value="Individual">
            <xsd:annotation>
                <xsd:documentation>Individual</xsd:documentation>
            </xsd:annotation>
        </xsd:enumeration>
    </xsd:restriction>
```

```
</xsd:simpleType>
<xsd:complexType name="PassportCategoryCodeType">
    <xsd:annotation>
        <xsd:documentation>
            A data type for a kind of passport.
        </xsd:documentation>
        <xsd:appinfo>
            <i:Base i:namespace="http://niem.gov/niem/structures/2.0"
                    i:name="Object"/>
        </xsd:appinfo>
    </xsd:annotation>
    <xsd:simpleContent>
        <xsd:extension base="nc:PassportCategoryCodeSimpleType">
            <xsd:attributeGroup ref="s:SimpleObjectAttributeGroup"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
```

2.6 **NIEMCodeValue**

Description

The NIEMCodeValue stereotype is a specialization of EnumerationLiteral. It represents a NIEM code value, which is implemented in XML Schema by an enumeration facet. See Section 9.12.3 of **[NIEM-NDR]** and Section 4.3.5 of **[XMLSchemaDatatypes]**.

Attributes and Associations

NIEMCodeValue inherits the following attribute from EnumerationLiteral; it shall map to an analogous NIEM concept and XML Schema implementation as described:

`name`: String [0..1]

> The NIEM code value, implemented in XML Schema as the value of the attribute `value` on the element `xsd:enumeration`, the child of the element `xsd:restriction`, which is the child of the element `xsd:simpleType`. See Section 4.3.5 of **[XMLSchemaDatatypes]**.

NIEMCodeValue defines the following attribute, which shall map to an analogous NIEM concept and XML Schema implementation as described:

`definition`: String [0..1]

> The data definition for the NIEM code value, implemented in XML Schema as the content of the first element `xsd:documentation` on the element `xsd:enumeration`. See Section 7.2.1 of **[NIEM-NDR]**, Rule 7-8.

Constraints

NIEMCodeValue does not adhere to any additional constraints.

Examples

See NIEMCodeSimpleType and NIEMCodeType.

2.7 NIEMCCCType

Description

The NIEMCCCType stereotype is a specialization of Class. It represents a NIEM type which is implemented in XML Schema as a complex type definition with complex content. See Section 7.4 of **[NIEM-NDR]** and Section 3.4 of **[XMLSchemaStructures]**.

NIEMCCCType is abstract; its specializations are NIEMObjectType, NIEMAssociationType, NIEMMetadataType, NIEMAugmentationType and NIEMAdapterType.

The acronym CCC in the stereotype name derives from the implementation of the NIEM concept, a complex type definition with complex content.

Attributes and Associations

NIEMCCCType inherits the following attribute from Class; it shall map to an analogous NIEM concept and XML Schema implementation as described:

`name`: String [0..1]

> The name of the NIEM type, implemented in XML Schema as the value of the attribute `name` on the element `xsd:complexType`.

NIEMCCCType defines the following attribute, which shall map to an analogous NIEM concept and XML Schema implementation as described:

`definition`: String [0..1]

> The data definition for the NIEM type, implemented in XML Schema as the content of the first element `xsd:documentation` on the element `xsd:complexType`. See Section 7.2.1 of **[NIEM-NDR]**, Rule 7-4.

Constraints

Specializations of NIEMCCCType define the constraints to which they adhere.

Examples

See NIEMObjectType, NIEMAssociationType, NIEMMetadataType, NIEMAugmentationType, and NIEMAdapterType.

2.8 NIEMObjectType

Description

The NIEMObjectType stereotype is a specialization of NIEMCCCType. It represents a NIEM object type which is implemented in XML Schema as a complex type definition with complex content. See Section 7.4.1 of **[NIEM-NDR]** and Section 3.4 of **[XMLSchemaStructures]**.

Attributes and Associations

NIEMObjectType has no attributes or associations that have an analogous NIEM concept beyond those inherited from NIEMCCCType.

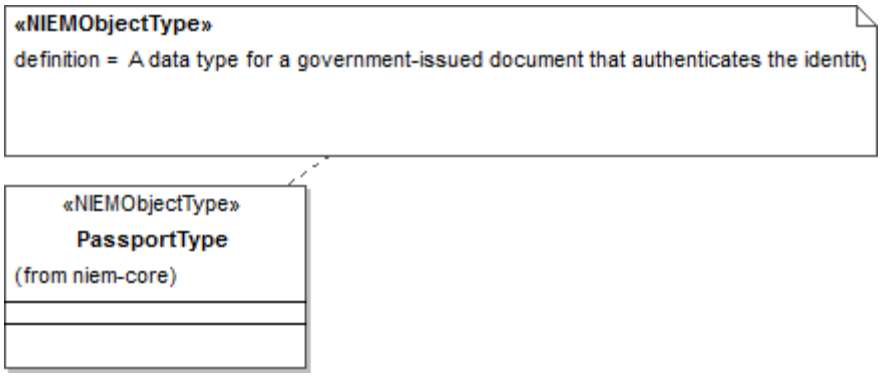
Constraints

NIEMObjectType adheres to the following constraints:

1. If the NIEMObjectType is the specific Classifier in a Generalization,
    a. the general Classifier must be a NIEMObjectType and
    b. the base type definition for the complex type definition that is the implementation of the NIEM object type shall be the type definition that is the implementation of the NIEM object type represented by the general Classifier.
2. If the NIEMObjectType is not the specific Classifier in a Generalization, the base type definition for the complex type definition shall be s:ComplexObjectType.

Examples

The following diagram illustrates the use of the stereotype:



The above diagram is implemented in XML Schema as follows:

```
<xsd:complexType name="PassportType">
    <xsd:annotation>
        <xsd:documentation>
            A data type for a government-issued document that authenticates
            the identity and citizenship of a person.
        </xsd:documentation>
        <xsd:appinfo>
            <i:Base i:namespace="http://niem.gov/niem/structures/2.0"
                    i:name="Object"/>
        </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
        <xsd:extension base="s:ComplexObjectType">
            <xsd:sequence>
                ...
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

2.9 **NIEMAssociationType**

Description

The NIEMAssociationType stereotype is a specialization of NIEMCCCType. It represents a NIEM association, type which is implemented in XML Schema as a complex type definition with complex content. See Section 7.4.3 of **[NIEM-NDR]** and Section 3.4 of **[XMLSchemaStructures]**.

Attributes and Associations

NIEMAssociationType has no attributes or associations that have an analogous NIEM concept beyond those inherited from NIEMCCCType.

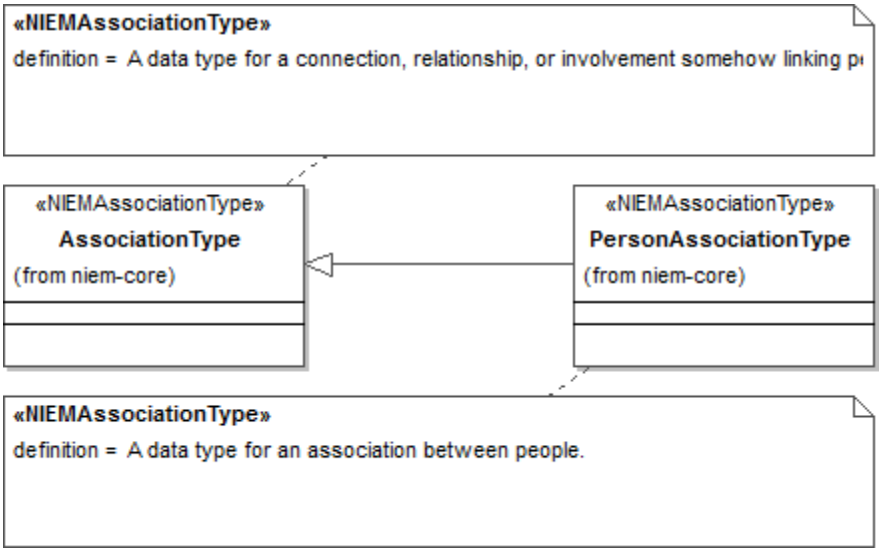
Constraints

NIEMAssociationType adheres to the following constraints:

1. If the NIEMAssociationType is the specific Classifier in a Generalization,
    a. the general Classifier must be a NIEMAssociationType and
    b. the base type definition for the complex type definition that is the implementation of the NIEM association type shall be the type definition that is the implementation of the NIEM association type represented by the general Classifier.
2. If the NIEMAssociationType is not the specific Classifier in a Generalization, the base type definition for the complex type definition shall be s:ComplexObjectType.

Examples

The following diagram illustrates the use of the stereotype:



The above diagram is implemented in XML Schema as follows:

```
<xsd:complexType name="AssociationType">
    <xsd:annotation>
        <xsd:documentation>
            A data type for a connection, relationship, or involvement
            somehow linking people and/or things together.
        </xsd:documentation>
        <xsd:appinfo>
            <i:Base i:namespace="http://niem.gov/niem/structures/2.0"
                    i:name="Association"/>
        </xsd:appinfo>
    </xsd:annotation>
```

```
    <xsd:complexContent>
        <xsd:extension base="s:ComplexObjectType">
            <xsd:sequence>
                ...
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="PersonAssociationType">
    <xsd:annotation>
        <xsd:documentation>
            A data type for an association between people.
        </xsd:documentation>
        <xsd:appinfo>
            <i:Base i:name="AssociationType"/>
        </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
        <xsd:extension base="nc:AssociationType">
            <xsd:sequence>
                ...
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

2.10 **NIEMMetadataType**

Description

The NIEMMetadataType stereotype is a specialization of NIEMCCCType. It represents a NIEM metadata type, which is implemented in XML Schema as a complex type definition with complex content. See Section 7.4.4 of **[NIEM-NDR]** and Section 3.4 of **[XMLSchemaStructures]**.

Attributes and Associations

NIEMMetadataType has no attributes or associations that have an analogous NIEM concept beyond those inherited from NIEMCCCType.

Constraints

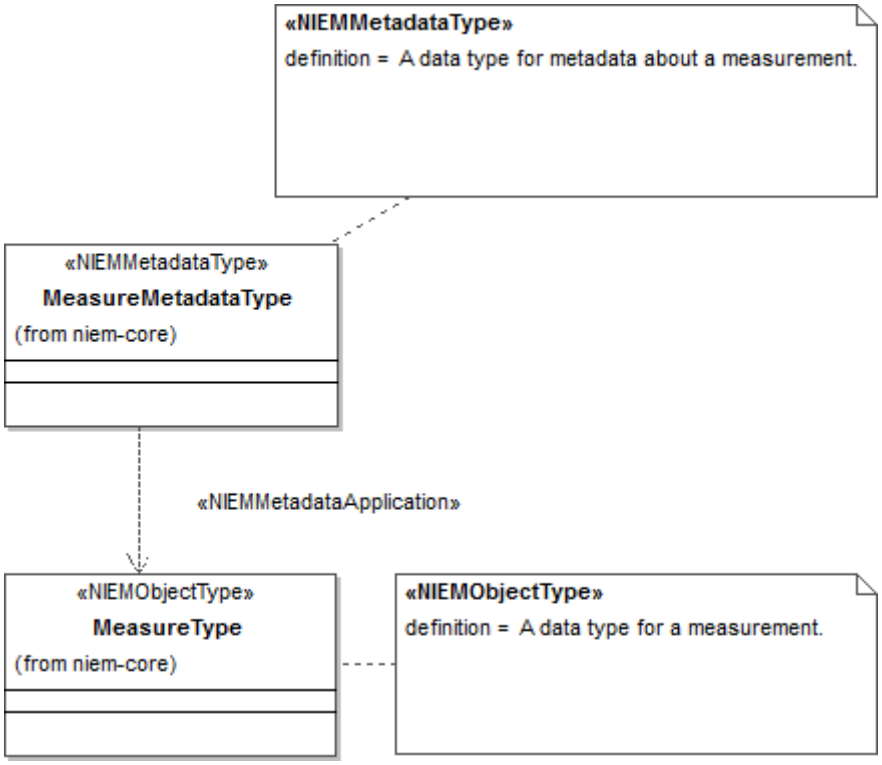
NIEMMetadataType adheres to the following constraints:

1. If the NIEMMetadataType is the specific Classifier in a Generalization,
    a. the general Classifier must be a NIEMMetadataType and
    b. the base type definition for the complex type definition that is the implementation of the NIEM metadata type shall be the type definition that is the implementation of the NIEM metadata type represented by the general Classifier.

2. If the NIEMMetadataType is not the specific Classifier in a Generalization, the base type definition for the complex type definition shall be s:MetadataType.

Examples

The following diagram illustrates the use of the stereotype:



The above diagram is implemented in XML Schema as follows:

```
<xsd:complexType name="MeasureType">
    <xsd:annotation>
        <xsd:documentation>A data type for a measurement.</xsd:documentation>
        <xsd:appinfo>
            <i:Base i:namespace="http://niem.gov/niem/structures/2.0"
                    i:name="Object"/>
        </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
        <xsd:extension base="s:ComplexObjectType">
            <xsd:sequence>
                ...
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="MeasureMetadataType">
    <xsd:annotation>
        <xsd:documentation>
```

```
            A data type for metadata about a measurement.
        </xsd:documentation>
        <xsd:appinfo>
            <i:Base i:namespace="http://niem.gov/niem/structures/2.0"
                    i:name="MetadataType"/>
            <i:AppliesTo i:name="MeasureType"/>
        </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
        <xsd:extension base="s:MetadataType">
            <xsd:sequence>
                ...
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

2.11 **NIEMMetadataApplication**

Description

The NIEMMetadataApplication stereotype is a specialization of Dependency. It represents a constraint on a NIEM metadata type that limits the applicability of the NIEM metadata type to specific types, which is implemented in XML Schema by application information. See Sections 7.2.2 and 7.4.4 of **[NIEM-NDR]**.

Attributes and Associations

NIEMMetadataApplication inherits the following associations from Dependency; they shall map to an analogous NIEM concept and XML Schema implementation as described:

`client` NamedElement 1 1

The reference to the NIEM metadata type.

`supplier` NamedElement 1 1

The reference to the NIEM type to which the NIEM metadata type applies, implemented in XML Schema as the value of the attribute `name` on the element `i:AppliesTo`.

Constraints

NIEMMetadataApplication adheres to the following constraints:

1. The attribute `client` must be a NIEMMetadataType.
2. The attribute `supplier` must be a NIEMObjectType.

Examples

See NIEMMetadataType.

2.12 NIEMAugmentationType

Description

The NIEMAugmentationType stereotype is a specialization of NIEMCCCType. It represents a NIEM augmentation type, which is implemented in XML Schema as a complex type definition with complex content. See Section 7.4.5 of **[NIEM-NDR]** and Section 3.4 of **[XMLSchemaStructures]**.

Attributes and Associations

NIEMAugmentationType has no attributes or associations that have an analogous NIEM concept beyond those inherited from NIEMCCCType.

Constraints

NIEMAugmentationType adheres to the following constraints:

1. If the NIEMAugmentationType is the specific Classifier in a Generalization,
    a. the general Classifier must be a NIEMAugmentationType and
    b. the base type definition for the complex type definition that is the implementation of the NIEM augmentation type shall be the type definition that is the implementation of the NIEM metadata type represented by the general Classifier.
2. If the NIEMAugmentationType is not the specific Classifier in a Generalization, the base type definition for the complex type definition shall be s:AugmentationType.

Examples

See NIEMProperty.

2.13 NIEMAdapterType

Description

The NIEMAdapterType stereotype is a specialization of NIEMCCCType. It represents a NIEM adapter type, which is implemented in XML Schema as a complex type definition with complex content. See Section 7.7 of **[NIEM-NDR]** and Section 3.4 of **[XMLSchemaStructures]**.

Attributes and Associations

NIEMAdapterType has no attributes or associations that have an analogous NIEM concept beyond those inherited from NIEMCCCType.

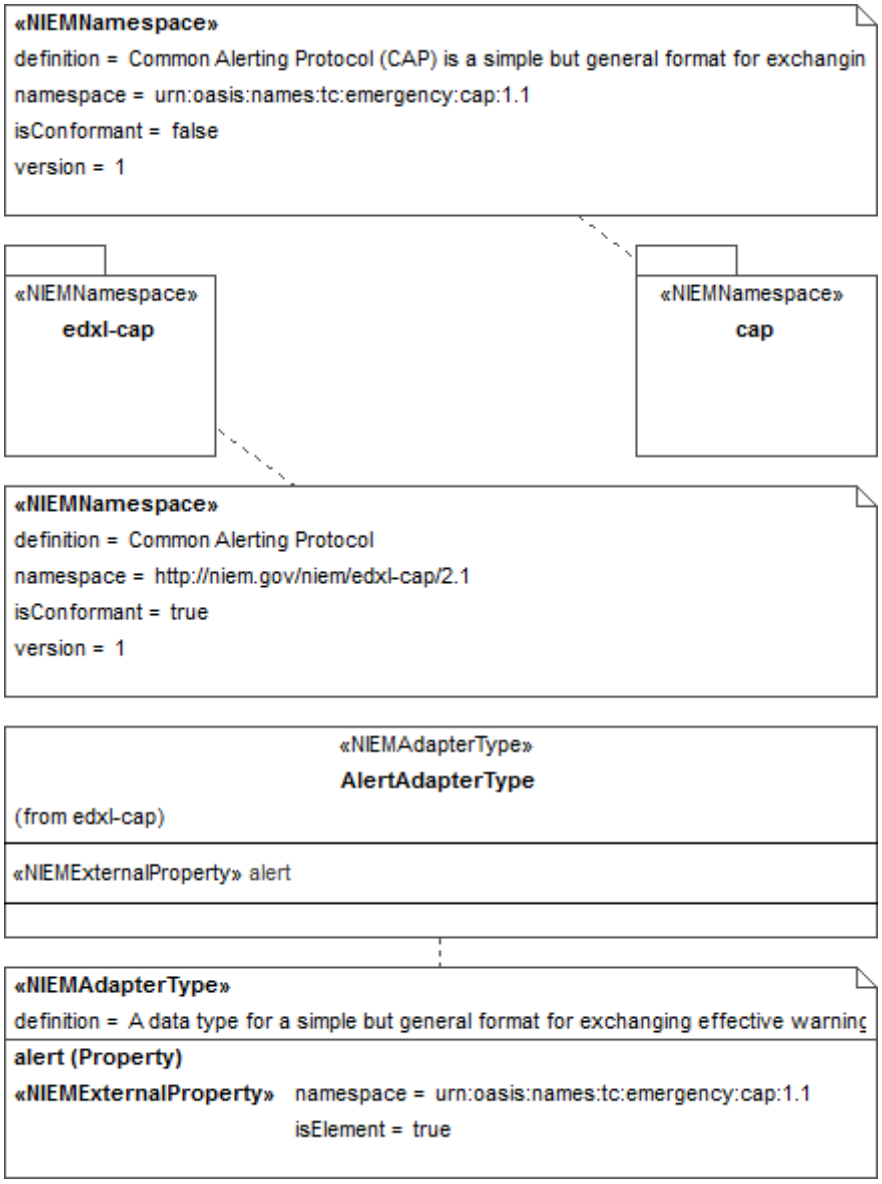
Constraints

NIEMAdapterType adheres to the following constraint:

1. NIEMAdapterType must not be the general or specific Classifier in a Generalization.

Examples

The following diagram illustrates the use of the stereotype:



The above diagram is implemented in XML Schema as follows:

```
<xsd:schema targetNamespace="http://niem.gov/niem/edxl-cap/2.1"
            version="1"
            xmlns:s="http://niem.gov/niem/structures/2.0"
            xmlns:cap="urn:oasis:names:tc:emergency:cap:1.1"
            xmlns:edxl-cap="http://niem.gov/niem/edxl-cap/2.1"
            xmlns:i="http://niem.gov/niem/appinfo/2.0">
    <xsd:annotation>
        <xsd:documentation>Common Alerting Protocol</xsd:documentation>
        <xsd:appinfo>
            <i:ConformantIndicator>true</i:ConformantIndicator>
        </xsd:appinfo>
    </xsd:annotation>
    <xsd:import schemaLocation="../../structures/2.0/structures.xsd"
                namespace="http://niem.gov/niem/structures/2.0"/>
```

```
    <xsd:import schemaLocation="../../appinfo/2.0/appinfo.xsd"
                namespace="http://niem.gov/niem/appinfo/2.0"/>
    <xsd:import schemaLocation="../../external/cap/1.1/cap.xsd"
                namespace="urn:oasis:names:tc:emergency:cap:1.1">
        <xsd:annotation>
            <xsd:documentation>
                Common Alerting Protocol (CAP) is a simple but general format
                for exchanging effective warning messages based on best
                practices identified in academic research and real-world
                experience.
            </xsd:documentation>
            <xsd:appinfo>
                <i:ConformantIndicator>false</i:ConformantIndicator>
            </xsd:appinfo>
        </xsd:annotation>
    </xsd:import>
    <xsd:complexType name="AlertAdapterType">
        <xsd:annotation>
            <xsd:documentation>
                A data type for a simple but general format for exchanging
                effective warning messages based on best practices identified
                in academic research and real-world experience.
            </xsd:documentation>
            <xsd:appinfo>
                <i:Base i:namespace="http://niem.gov/niem/structures/2.0"
                        i:name="Object"/>
                <i:ExternalAdapterTypeIndicator>
                    true
                </i:ExternalAdapterTypeIndicator>
            </xsd:appinfo>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="s:ComplexObjectType">
                <xsd:sequence>
                    <xsd:element ref="cap:alert"
                                 minOccurs="0"
                                 maxOccurs="unbounded"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:schema>
```

2.14 **NIEMExternalProperty**

Description

The NIEMExternalProperty stereotype is a specialization of Property. It represents an external component, which is implemented in XML Schema as either an attribute use or a particle. See Section 7.7 of **[NIEM-NDR]** and Sections 3.2 and 3.3 of **[XMLSchemaStructures]**.

Attributes and Associations

NIEMExternalProperty inherits the following attributes from Property; they shall map to an analogous NIEM concept and XML Schema implementation as described:

`name`: String [0..1]

> The name of the NIEM property, implemented in XML Schema as name property of the attribute or element declaration; the value of the attribute `name` on the element `xsd:attribute` or `xsd:element`.

`lower`: Integer [0..1]

> Implemented in XML Schema as the min occurs property of the particle; the value of the attribute `minOccurs` on the element `xsd:element`. Default is "0".

`upper`: UnlimitedNatural [0..1]

> Implemented in XML Schema as the max occurs property of the particle; the value of the attribute `maxOccurs` on the element `xsd:element`. Default is "*".

NIEMExternalProperty defines the following attributes, which shall map to an analogous NIEM concept and XML Schema implementation as described:

`isElement`: Boolean [0..1]

> Indicates whether the external component is implemented in XML Schema as an attribute use or particle. Default is "true".

`namespace`: String [1..1]

> The namespace of the external component.

Constraints

NIEMExternalProperty adheres to the following constraints:

1. NIEMExternalProperty must be the ownedAttribute of a NIEMAdapterType.
2. If the attribute isElement is false, the value of the attributes lower and upper must be "1" or the attributes lower and upper must be empty.

Examples

See NIEMAdapterType.

2.15 **NIEMProperty**

Description

The NIEMProperty stereotype is a specialization of Property. It represents a NIEM property, which is implemented in XML Schema as either an attribute declaration and attribute use or an element declaration and particle. See Sections 3.2 and 3.3 of **[XMLSchemaStructures]**.

There are significant differences between the UML representation and XML Schema implementation of a NIEM property. Sections 6.1.6.2 and 6.1.6.3 of **[NIEM-NDR]**, Rule 6-14 and Rule 6-15, require that an attribute or element declaration be a top-level declaration; however, Section 7.3.44 of **[UMLSuperstructure]** requires that a Property be the `ownedAttribute` of a Classifier. Thus in the UML representation, only one Classifier may reference a Property, while in the XML Schema implementation, more than one type definition may reference the same attribute or element declaration.

To resolve this difference, more than one NIEMProperty with the same values for attributes `name` and `namespace` shall have the same attribute or element declaration. The first NIEMProperty defines both the attribute or element declaration and the attribute use or particle that reference that declaration; the rest define only the attribute use or particle that references that declaration.

Attributes and Associations

NIEMProperty inherits the following attributes from Property; they shall map to an analogous NIEM concept and XML Schema implementation as described:

`name`: String [0..1]

> The name of the NIEM property, implemented in XML Schema as name property of the attribute or element declaration; the value of the attribute `name` on the element `xsd:attribute` or `xsd:element`.

`lower`: Integer [0..1]

> Implemented in XML Schema as the min occurs property of the particle; the value of the attribute `minOccurs` on the element `xsd:element`. Default is "0".

`type`: Type [0..1]

> The type of the NIEM property. If present, implemented in XML Schema as the type definition property of the attribute or element declaration; the value of the attribute `type` on the element `xsd:attribute` or `xsd:element`. If absent, implemented in XML Schema by the abstract property of the element declaration; the value "true" of the attribute `abstract` on the element `xsd:element`.

`upper`: UnlimitedNatural [0..1]

Implemented in XML Schema as the max occurs property of the particle; the value of the attribute `maxOccurs` on the element `xsd:element`. Default is "*".

NIEMProperty defines the following attributes, which shall map to an analogous NIEM concept and XML Schema implementation as described:

`augmentedTypeName`: String [0..1]

If the NIEM property is an augmentation, the NIEM type to which the augmentation applies, implemented in XML Schema as the value of the attribute `name` on the element `i:AppliesTo`.

`augmentedTypeNamespace`: String [0..1]

If the NIEM property is an augmentation, the namespace of the NIEM type to which the NIEM augmentation applies, implemented in XML Schema as the value of the attribute `namespace` on the element `i:AppliesTo`.

`definition`: String [0..1]

The data definition for the NIEM property, implemented in XML Schema as the content of the first element `xsd:documentation` on the element `xsd:attribute` or `xsd:element`. See Section 7.2.1 of **[NIEM-NDR]**, Rule 7-6 and Rule 7-7.

`isElement`: Boolean [0..1]

Indicates whether the NIEM property is implemented in XML Schema as an attribute declaration and attribute use or element declaration and particle. Default is "true".

`nillable`: Boolean [0..1]

Implemented in XML Schema as the nillable property of the element declaration; the value of the attribute `nillable` on the element `xsd:element`. Default is "false".

`namespace`: String [0..1]

The namespace of the NIEM property.

`substitutionGroupName`: String [0..1]

Implemented in XML Schema as the substitution group affiliation property of the element declaration; the value of the attribute `substitutionGroup` on the element `xsd:element`.

`substitutionGroupNamespace`: String [0..1]

The namespace of the substitution group.

`value`: NIEMPropertyValueCodeType [0..1]

Indicates the method by which the model associates the element and its value: "content", "contentMayBeReference", "reference", or "referenceMayBeContent".

- "content" indicates that the model associates the element with its value as content;
- "contentMayBeReference" indicates that the model associates the element with its value as content in the context of this type, but may associate the element with its value as reference in another;
- "reference" indicates that the model associates the element with its value as reference; and
- "referenceMayBeContent" indicates that the model associates the element with its value as reference in the context of this type, but may associate the element with its value as content in another.

Default is "contentMayBeReference".

In the context of NIEMTopLevel, "contentMayBeReference" and "referenceMayBeContent" have the same meaning; both indicate that the model may associate the element with its value as content or as reference.

Constraint

NIEMProperty adheres to the following constraints:

1. NIEMProperty must be the `ownedAttribute` of a NIEMSimpleType, NIEMCSCType, NIEMCodeSimpleType, NIEMCodeType, or NIEMCCCType.
2. If the attribute `namespace` is empty, the namespace for the attribute or element declaration that is the implementation of the NIEM property shall be the same as the namespace for the type definition that is the implementation of the NIEM type represented by its owner NIEMSimpleType, NIEMCSCType, NIEMCodeSimpleType, NIEMCodeType, or NIEMCCCType.
3. If the attribute `substitutionGroupName` is not empty and the attribute `substitutionGroupNamespace` is empty, the namespace of the substitution group shall be the same as the namespace for the attribute or element declaration that is the implementation of the NIEM property.
4. If the attribute `isElement` is false and the attribute `type` is empty,

a. the value of the attributes `lower` and `upper` must be "1" or the attributes `lower` and `upper` must be empty;

b. the attributes `substitutionGroupName` and `substitutionGroupNamespace` must be empty;

c. the value of the attribute `value` must be "content"; and

d. type definition of the attribute declaration that is the implementation of the NIEM property shall be xsd:string.

5. If the attribute `isElement` is false and the attribute `type` is not empty,

a. the value of the attributes `lower` and `upper` must be "1" or the attributes `lower` and `upper` must be empty;

b. the attributes `substitutionGroupName` and `substitutionGroupNamespace` must be empty;

c. the value of the attribute `value` must be "content";

d. the attribute `type` must be a NIEMSimpleType; and

e. type definition of the attribute declaration that is the implementation of the NIEM property shall be the type definition that is the implementation of the NIEM type represented by the attribute `type`.

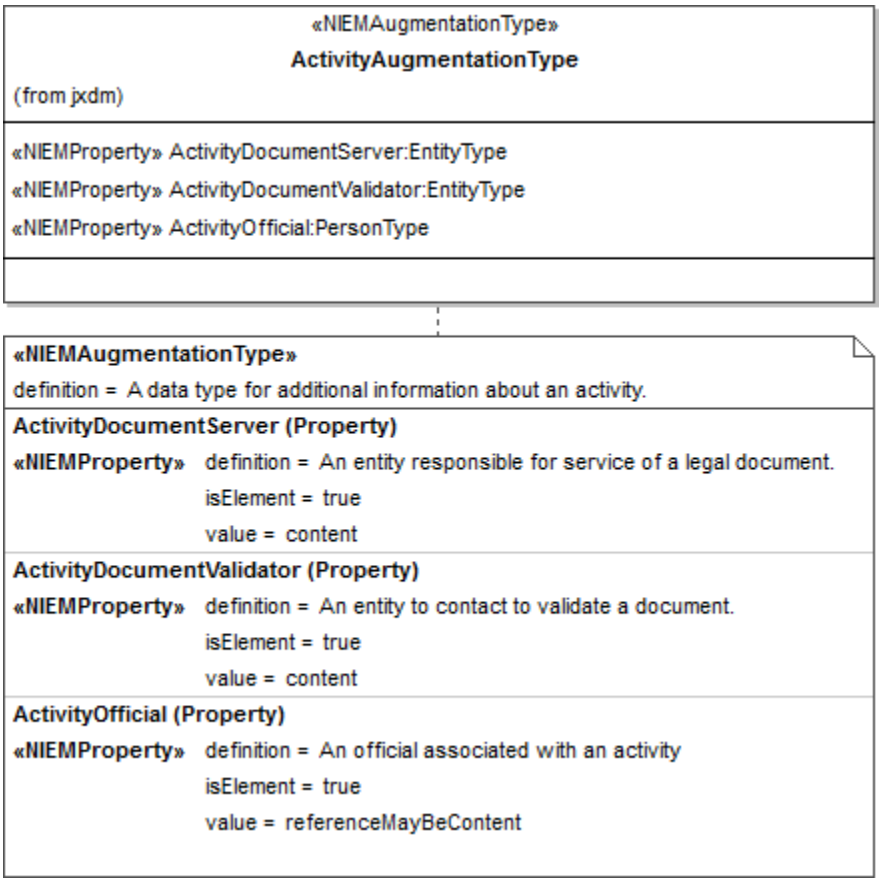
6. If the attribute `isElement` is empty or true and the attribute `type` is not empty,

a. the attribute `type` must be a NIEMCSCType or NIEMCCCType and

b. type definition of the element declaration that is the implementation of the NIEM property shall be the type definition that is the implementation of the NIEM type represented by the attribute `type`.

Examples

The following diagram illustrates the use of the stereotype:



The above diagram is implemented in XML Schema as follows:

```
<xsd:complexType name="ActivityAugmentationType">
    <xsd:annotation>
        <xsd:documentation>
            A data type for additional information about an activity.
        </xsd:documentation>
        <xsd:appinfo>
            <i:Base i:namespace="http://niem.gov/niem/structures/2.0"
                    i:name="AugmentationType"/>
        </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
        <xsd:extension base="s:AugmentationType">
            <xsd:sequence>
                <xsd:element ref="j:ActivityDocumentServer"
                             minOccurs="0"
                             maxOccurs="unbounded"/>
                <xsd:element ref="j:ActivityDocumentValidator"
                             minOccurs="0"
                             maxOccurs="unbounded"/>
                <xsd:element ref="j:ActivityOfficialReference"
                             minOccurs="0"
                             maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:extension>
```

```
    </xsd:complexContent>
</xsd:complexType>
<xsd:element name="ActivityDocumentServer"
             type="nc:EntityType"
             nillable="true">
    <xsd:annotation>
        <xsd:documentation>
            An entity responsible for service of a legal document.
        </xsd:documentation>
    </xsd:annotation>
</xsd:element>
<xsd:element name="ActivityDocumentValidator"
             type="nc:EntityType"
             nillable="true">
    <xsd:annotation>
        <xsd:documentation>
            An entity to contact to validate a document.
        </xsd:documentation>
    </xsd:annotation>
</xsd:element>
<xsd:element
    name="ActivityOfficialReference"
    type="s:ReferenceType">
    <xsd:annotation>
        <xsd:documentation>
            An official associated with an activity
        </xsd:documentation>
        <xsd:appinfo>
            <i:ReferenceTarget
                 i:namespace="http://niem.gov/niem/niem-core/2.0"
                 i:name="PersonType"/>
        </xsd:appinfo>
    </xsd:annotation>
</xsd:element>
<xsd:element name="ActivityOfficial"
             type="nc:PersonType"
             nillable="true">
    <xsd:annotation>
        <xsd:documentation>
            An official associated with an activity
        </xsd:documentation>
    </xsd:annotation>
</xsd:element>
```

2.16 **NIEMTopLevel**

Description

The NIEMTopLevel stereotype is a specialization of Class. It does not represent any NIEM concept; it exists to permit the user to define a NIEM property that is not used by any NIEM type.

There are significant differences between the UML representation and XML Schema implementation of a NIEM property. Sections 6.1.6.2 and 6.1.6.3 of **[NIEM-NDR]**, Rule 6-14

and Rule 6-15, require that an attribute or element declaration be a top-level declaration, but **[NIEM-NDR]** does not require a corresponding attribute use or particle; however, Section 7.3.44 of **[UMLSuperstructure]** requires that a Property be the `ownedAttribute` of a Classifier. Thus in the UML representation, the declaration and use of a Property are not distinct, and the declaration of a Property requires its use. In the XML Schema implementation, the declaration and use are distinct, and the declaration does not require a corresponding use.

To resolve this difference, any NIEMProperty within a NIEMTopLevel shall represent an attribute or element declaration without a correpsonding attribute use or particle.

Attributes and Associations

NIEMTopLevel has no attributes or associations that have an analogous NIEM concept.

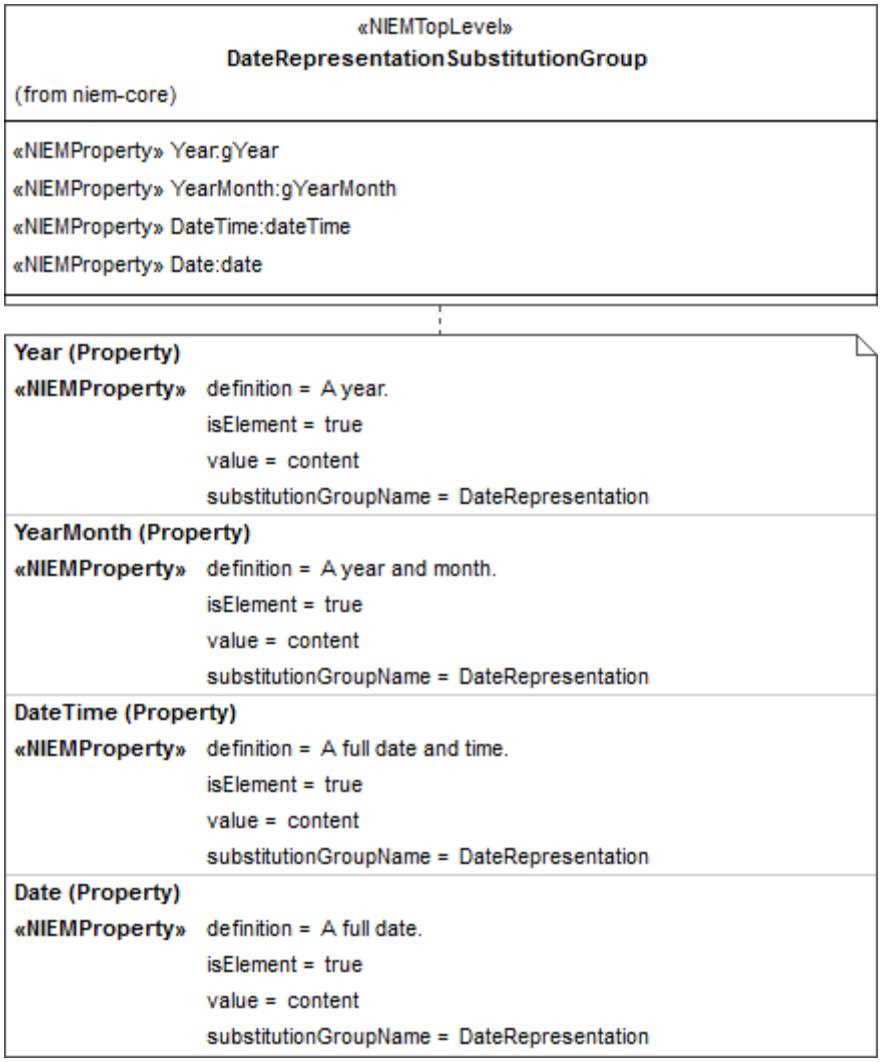
Constraint

NIEMTopLevel adheres to the following constraint:

> NIEMTopLevel must not be the general or specific Classifier in a Generalization.

Examples

The following diagram illustrates the use of the stereotype:



The above diagram is implemented in XML Schema as follows:

```
<xsd:element
    substitutionGroup="nc:DateRepresentation"
    name="Year"
    type="niem-xsd:gYear">
    <xsd:annotation>
      <xsd:documentation>A year.</xsd:documentation>
      <xsd:appinfo>
        <i:Base i:name="DateRepresentation"/>
      </xsd:appinfo>
    </xsd:annotation>
</xsd:element>

<xsd:element
    substitutionGroup="nc:DateRepresentation"
    name="YearMonth"
    type="niem-xsd:gYearMonth">
    <xsd:annotation>
      <xsd:documentation>A year and month.</xsd:documentation>
```

```
      <xsd:appinfo>
        <i:Base i:name="DateRepresentation"/>
      </xsd:appinfo>
    </xsd:annotation>
</xsd:element>

<xsd:element
    substitutionGroup="nc:DateRepresentation"
    name="DateTime"
    type="niem-xsd:dateTime">
    <xsd:annotation>
      <xsd:documentation>A full date and time.</xsd:documentation>
      <xsd:appinfo>
        <i:Base i:name="DateRepresentation"/>
      </xsd:appinfo>
    </xsd:annotation>
</xsd:element>

<xsd:element
    substitutionGroup="nc:DateRepresentation"
    name="Date"
    type="niem-xsd:date">
    <xsd:annotation>
      <xsd:documentation>A full date.</xsd:documentation>
      <xsd:appinfo>
        <i:Base i:name="DateRepresentation"/>
      </xsd:appinfo>
    </xsd:annotation>
</xsd:element>
```