TYPE AUGMENTATION SUPPLEMENT TO NDR 1.3

VERSION 1.0
URI: http://reference.niem.gov/niem/specification/naming-and-design-rules/1.3/type-augmentation/

⟨NIEM⟩

Change History

No.	Date	Reference: All, Page, Table, Figure, Paragraph	A = Add. M = Mod. D = Del.	Revised By	Change Description
1.0	24/09/2009	All	A	NTAC	Initial version

Contents

1 Introduction

The NIEM Technical Architecture Committee (NTAC) defined *type augmentation* in the NIEM Naming and Design Rules (NDR) [Ref 1] to establish a method by which a domain could identify properties that apply to a type not defined in the domain. This document describes the manner in which augmentations may be applied within a domain schema to support that domain's Information Exchange Package Documentation (IEPD) developers. It defines augmented types and augmented elements, which are domain-specific data components designed to support a single-domain IEPD. A single-domain IEPD uses only data components drawn from the NIEM Core and the NIEM domain for which it is developed. A multi-domain IEPD uses data components from NIEM Core and two or more NIEM domains.

This document is considered a normative specification that supplements NIEM NDR 1.3.

2 Background

In 2006 the NTAC identified a problem with the creation of subclasses from Core types. This was a result of the multi-domain nature of NIEM (unlike the single domain nature of Global Justice XML Data Model). Within its own namespace, each NIEM domain could independently create a different subclass of the same Core type through `xsd:extension`. This made it impossible for IEPD developers to reuse the elements typed by those subclasses (subtypes) in combination. The properties added through extension by each domain were locked in each domain's version of the subclass.

To illustrate:

Domain d1 extends `nc:OrgType` with properties `d1:A` and `d1:B` resulting in `d1:OrgType`. Now `d1:OrgType` contains all the properties of the base, `nc:OrgType` plus d1:A and d1:B. Then d1 creates `d1:Org` of type `d1:OrgType`.

Likewise, domain d2 extends `nc:OrgType` with properties `d2:C` and `d2:D` resulting in `d2:OrgType`. Now `d2:OrgType` contains all the properties of the base, `nc:OrgType` plus `d2:C` and `d2:D`. Then d2 creates `d2:Org` of type `d2:OrgType`.

The above scenario results in the following types and associated elements:

Types	Corresponding Elements

`nc:OrgType`
    `nc:X`
    `nc:Y`
	`nc:Org` (of type `nc:OrgType`)

`d1:OrgType`
    `nc:X`
    `nc:Y`
    `d1:A`
    `d1:B`
	`d1:Org` (of type `d1:OrgType`)

```
d2:OrgType              d2:Org (of type d2:OrgType)
    nc:X
    nc:Y
    d2:C
    d2:D
```

In such cases, properties were added to supplement the original base type (`nc:OrgType`) and NOT to create an actual subclass which must have uniquely specialized semantics, must be mutually exclusive of other potential subclasses of this base type, and must be a persistent data concept (i.e., not temporary). As a result, an IEPD developer cannot reuse `d1:Org` and `d2:Org` together in any useful way since the additional properties are locked in two different types. The only alternative is to rebuild a completely new type using all 6 properties X, Y, A, B, C, and D. Of course, this results in still another `iepd:OrgType` which is semantically indistinguishable from the other three OrgTypes (except for various combinations of the six elements). This problem is closely related to multiple inheritance, which is not supported by XML Schema.

To resolve this problem, the NTAC evaluated 11 different techniques that enable effective reuse of properties in combination across domains when subclasses are not appropriate. One technique was selected as the current method, referred to as *Type Augmentation*. Note that this technique was designed to resolve a problem that occurs when IEPD developers must use content from multiple domains that are allowed to develop NIEM content for namespaces independently.

Type augmentation allows a domain to create augmentation types and associated elements (containers of properties), that can be reused in combination to construct new types in IEPDs which are derived from a base type in Core or another domain. Furthermore, an IEPD developer can place an element created from an augmented type in the substitution group of the base element so that it can substitute for the base element anywhere it occurs in an IEPD instance.

Since the application of augmentation elements (by extending a base type) results in another type that is not semantically distinct from its base type, application and use of augmentations previously have been restricted to IEPDs [Ref 2]. Applying and using augmentation elements (containers) within domain namespaces could increase confusion to IEPD developers building multi-domain IEPDs. Domains would potentially contain multiple types with little or no semantic distinction and these types cannot not be used effectively by IEPD developers building multi-domain IEPDs.

Recently during the preparation of the NIEM minor release 2.1, several NIEM domains requested they be allowed to apply augmentation elements to new types within their domain namespaces. This document supplements NIEM NDR 1.3 and represents a compromise to the previous restriction on the application of augmentation elements within domain namespaces.

3 Definitions

The following terms are used in subsequent sections of this specification:

1. *domain D* – a schema for a specific NIEM domain, which defines and uses augmentations.

2. *base type* – a type not defined by domain D, to which an augmentation type applies. The base type is identified by "applies to" metadata of an augmentation type.
3. *augmentation type* – a type defined by domain D, which lists a set of properties that apply to a base type, as specified by the NIEM NDR 1.3.
4. *augmentation element* – an element defined by domain D, which has a type that is an augmentation type, as specified by the NIEM NDR 1.3.
5. *base element* – an element that has a type that is the base type.
6. *augmented type* – A domain-created type that uses an augmentation element. An augmented type is an extension of the base type.
7. *augmented element* – A domain-created element that has a type that is an augmented type.

4 Modeling Rules

1. Domain D MAY:
   a. Define augmentation types that contain properties that apply to a particular base type.
   b. Define augmentation elements (containers), each with a type of an augmentation type.
   c. Define an augmented type that uses augmentation elements from domain D.
   d. Define any number of augmented elements, each with a type of an augmented type from domain D.
2. An augmented type defined by domain D MUST be defined as follows:
   a. An augmented type MUST use augmentation elements from domain D.
   b. An augmented type MUST be an immediate `xsd:extension` of its base type.
   c. Every augmentation element used by an augmented type MUST be applicable to the base type.
   d. A base type MUST NOT be defined by the domain D. But it MAY be defined by NIEM-Core or by another domain.
   e. An augmented type MUST NOT be referred to by any means other than as the type of an augmented element defined by domain D. An augmented type MUST NOT be referred to in any other way: not as a base for xsd:extension, nor as the type of any element that is not an augmented element.
   f. An augmented type must be annotated using appropriate NIEM `xsd:appinfo`.
3. An augmented element defined by domain D MUST be defined as follows:
   a. An augmented element MUST have a type that is its augmented type, also defined by domain D.
   b. An augmented element MUST be in the `substitutionGroup` of its base element.
   c. The base element MAY be referred to and used from within domain D.
   d. The base element MUST have a type that is the base type.
   e. The definition of an augmented element MUST be equivalent to that of its base element. The definition of an augmented element MUST NOT diverge from that of its base element, apart from having an extended type (the augmented type).

f. The name of an augmented element MUST be substantially similar to the name of its base element. The name of an augmented element MUST NOT diverge from that of its base element, apart from indicating that it is of an extended type (the augmented type).

g. An augmented element MUST NOT be referred to by any domain, including the defining domain D.

h. An augmented element must be annotated using appropriate NIEM `xsd:appinfo`.

5 Publication Area Rules

The NIEM High-Level Version Architecture [Ref 3] and Tool Architecture [Ref 4] define the *Publication Area* where domains may publish out-of-cycle updates to their own namespaces. A domain-managed schema in the Publication Area MAY use or reference data components from any domain, including augmented types and augmented elements.

6 Information Exchange Package Documentation (IEPD) Rules

1. An IEPD MAY use augmentation elements (containers) to construct new types.
2. An IEPD MAY use augmented types and augmented elements from domain schemas.
3. An IEPD SHOULD define its own types and elements (in an extension schema) when combining augmentations from multiple domains. This will help to provide consistency across IEPDs with respect to how augmentations from multiple domains are combined.

7 Naming and Design Rules Update

Sometime in the future, the NIEM Naming and Design Rules (NDR) 1.3 will be updated to a new version that incorporates and details the above guidelines and rules.

8 Identification (marking)

Because domain augmented types and augmented elements adhere to a different set of rules than do regular domain content, these components must be clearly identified (marked) for tools. The NIEM appinfo schema defines the metadata for the NIEM `xsd:appinfo` elements. However, because this schema is part of NIEM Core, it cannot be modified in release 2.1. Therefore, NIEM 2.1 will:

1. Contain a new additive appinfo schema that provides appropriate metadata elements for identifying augmented types and augmented elements.
2. The following elements will be added to appinfo.xsd (as a separate additive 2.1 schema that supplements the 2.0 schema) for augmented types and augmented elements:

```
<xsd:element name="AugmentedTypeIndicator" type="xsd:boolean">
```

```
      <xsd:annotation>
        <xsd:documentation>
            The AugmentedTypeIndicator may be applied to any
            NIEM-conformant type.  A type that occurs in a NIEM
            domain and that uses an augmentation element must
            have an AugmentedTypeIndicator with a true value.  No
            other type should have a true value
        </xsd:documentation>
      </xsd:annotation>
  </xsd:element>

  <xsd:element name="AugmentedElementIndicator"
                type="xsd:boolean">
      <xsd:annotation>
        <xsd:documentation>
            The AugmentedElementIndicator may be applied to any
            NIEM-conformant element.  An element that occurs in a
            NIEM domain and that has a type that is an augmented
            type should have a true value.  No other elements
            should have a true value
        </xsd:documentation>
      </xsd:annotation>
  </xsd:element
```

3. Import the new appinfo schema into 2.1 domain schemas, when required. The new appinfo schema will be imported along with the original appinfo schema used in NIEM 2.0. The NIEM 2.0 appinfo schema will not be modified, and the new appinfo schema will not overlap 2.0 appinfo content.
4. Annotate augmented types and augmented elements with appinfo elements from the new appinfo schema.

9 Schema Subset Generation Tool (SSGT) Behavior

SSGT (http://niem.gtri.gatech.edu/niemtools/ssgt/index.iepd) will be modified to incorporate the following behaviors:

1. Explicitly identify augmented types and elements in their respective definition Web pages.
2. Employ a search filter to allow users to include and exclude augmented types and elements in search.
3. Provide the new additive appinfo schema as part of any generated subset that contains augmented types or augmented elements.

10 After NIEM 2.1

After NIEM 2.1 has been released, and before another release of NIEM or update to NIEM NDR 1.3, the NTAC reserves the option to reconsider the aforementioned type augmentation rules and markings.

Appendix A: References

1. *NIEM Naming and Design Rules (NDR),* 31 October 2008, version 1.3, NIEM Technical Architecture Committee (NTAC), http://www.niem.gov/pdf/NIEM-NDR-1-3.pdf

2. *Techniques for Building and Extending NIEM XML Components,*7 August 2007, version 2.0.1, Georgia Tech Research Institute, http://www.niem.gov/topicIndex.php?topic=techPDF

3. *NIEM High-Level Version Architecture*, 21 July 2008, version 1.0, NIEM Technical Architecture Committee (NTAC), http://www.niem.gov/pdf/NIEM_HLVA.pdf

4. *NIEM High-Level Tool Architecture*, 1 December 2008, version 1.1, NIEM Technical Architecture Committee (NTAC), http://www.niem.gov/pdf/HLTA.pdf