**Simplified NIEM XML**

**Synopsis**

NIEM uses XML Schema to represent data models. Those schemas for data models are not convenient for data exchange developers. Now that we have CMF to represent data models, we can produce simpler schemas that make developers happy. We might also produce simpler messages, but that requires more thought.

**Discussion**

NIEM uses XML Schema to represent data models. NIEM Core, the NIEM domains, the data model in every message specification – all these are defined as an XML schema created by assembling a collection of XML schema documents. The NDR is the special sauce that is added to XML Schema to make this possible. Almost all of the rules in the NDR fall into one of these categories:

1. They define the semantics of NIEM XML schemas and NIEM XML data. For instance, subclasses are represented by xs:extension, and subproperties are represented through substitution groups.
2. They support schema reuse and composition. For instance, final="true" makes reuse more difficult, so it is not allowed in a reference schema.
3. They forbid XML schema features that do not work well with NIEM semantic interpretation. For instance, model groups and repeatable sequences do not have a useful interpretation in NIEM, and are therefore not allowed.
4. They forbid XML schema features that are not well supported in COTS tools, or were not well supported in the past. For instance, xs:include can cause problems in schema assembly, and is not interpreted consistently by different validating parsers.

Those rules allow NIEM XML schemas to do a fine job of representing data models. However, developers have a number of complaints when it comes to using those schemas to implement a data exchange. These complaints include:

1. There are many namespaces, and many schema documents that must be assembled correctly to form the schema.
2. There are many global elements. This results in very long drop-down menus in some IDEs. Also, the length of a message serialized as EXI can be slightly increased. Finally, it is not possible to determine the message root element from the schema.
3. The names of the elements follow ISO 11179 and use upper camel case. Many implementations are written in a programming language with a lower camel case convention. Developers would prefer shorter element names that make sense for their particular application.
4. There are many substitution groups. Many tool suites work better when xs:choice is used instead.
5. The schemas are not usable for XML validation in a cross-domain solution transfer agent (a "guard"). It is difficult to add string-length constraints to particular elements in order to fix this.

Now that we have the Common Model Format (CMF) and supporting tools, we do not have to use XML Schema to represent NIEM data models. We can represent those models in CMF, and use the tools to generate XML schema documents that are more suitable for data exchange implementations. Consider the following example message:

<hs:PersonOtherKinAssociation

xmlns:hs="http://release.niem.gov/niem/domains/hs/4.1/"

xmlns:j="http://release.niem.gov/niem/domains/jxdm/6.0/"

xmlns:nc="http://release.niem.gov/niem/niem-core/4.0/">

<hs:SourcePerson>

<nc:PersonAgeMeasure>

<nc:MeasureIntegerValue>14</nc:MeasureIntegerValue>

<nc:TimeUnitCode>ANN</nc:TimeUnitCode>

</nc:PersonAgeMeasure>

<j:PersonHairColorCode>BRO</j:PersonHairColorCode>

<nc:PersonName>

<nc:PersonGivenName>Rick</nc:PersonGivenName>

<nc:PersonSurName>Wilson</nc:PersonSurName>

</nc:PersonName>

</hs:SourcePerson>

<hs:TargetPerson>

<j:PersonHairColorCode>GRY</j:PersonHairColorCode>

<nc:PersonName>

<nc:PersonGivenName>Mortimer</nc:PersonGivenName>

<nc:PersonSurName>Smith</nc:PersonSurName>

<nc:PersonNameSuffixText>Sr</nc:PersonNameSuffixText>

<nc:PersonPreferredName>Morty</nc:PersonPreferredName>

</nc:PersonName>

</hs:TargetPerson>

<hs:PersonOtherKinAssociationCategoryCode>

Maternal Grandfather

</hs:PersonOtherKinAssociationCategoryCode>

</hs:PersonOtherKinAssociation>

What could we do to simplify the schema for this message without changing the message at runtime? Any or all of the following:

1. We could eliminate some of the global element declarations. (However, every element referenced in a namespace other than its own must remain global.)
2. We could replace substitution groups with xs:choice.
3. We could remove the attributes in structures:SimpleObjectAttributeGroup when the message specification does not use them. (Messages without IDREFs are much easier to process.)
4. We could replace long type derivation chains with the type at the end. For instance, nc:PersonGivenName has type nc:PersonNameTextType, which extends nc:ProperNameTextType, which extends nc:TextType. All that could be replaced by xs:string, extended by any attributes actually in use.

Easy enough to generate such a simplified schema from a CMF model. That would help with complaints #2, #4, and #5, and might look like this:

<xs:element name="PersonGivenName">

<xs:complexType>

<xs:complexContent>

<xs:extension base="xs:string">

<xs:attribute name="partialIndicator" type="xs:boolean" … />

<xs:attributeGroup ref="structures:SimpleObjectAttributeGroup"/>

And in most message specifications, which don't use IDREFs and don't include@partialIndicator, the schema would be even simpler:

<xs:element name="PersonGivenName" type="xs:string"/>

Those simplifications should be easy. They amount to automated support for constraint schemas. Since there is nothing really new in these simplifications, the impact is low, and there seems little reason not to implement them.

Now, if we are willing to simplify the message as well as the schema, then the following simplifications can be added to those listed above:

1. All of the schema declarations and definitions could be in a single namespace and schema document.
2. All global elements other than the message element could be made local.
3. ISO 11179 element and attribute names could be replaced with programmer-friendly names.

The simplified message might look like this:

<kinAssociation

xmlns="http://my/message/specification/URI/"

<sourcePerson>

<age>

<measureIntegerValue>14</measureIntegerValue>

<timeUnitCode>ANN</timeUnitCode>

</age>

<hairColorCode>BRO</hairColorCode>

<name>

<givenName>Rick</givenName>

<surName>Wilson</surName>

</name>

</sourcePerson>

<targetPerson>

<hairColorCode>GRY</hairColorCode>

<name>

<givenName>Mortimer</givenName>

<surName>Smith</surName>

<nameSuffixText>Sr</nameSuffixText>

<preferredName>Morty</preferredName>

</name>

</targetPerson>

<kinAssociationCategoryCode>

Maternal Grandfather

</kinAssociationCategoryCode>

</kinAssociation>

It will not be difficult to generate a schema for this message format from a CMF model. In addition, we will want a mapping from canonical NIEM component names to the programmer-friendly names. We would also generate XSLT to convert between the simplified message and the canonical message. This would help with all five of the programmer complaints listed.

It would seem that the simplified message gives up the self-describing nature of NIEM XML data. With existing NIEM messages, the namespace declarations plus qualified names combine to give us a URI for every component in the message. With those URIs we can hope to obtain the message specification and component definitions. Also, common practice makes many components recognizable on sight; for example, any NIEM developer understands nc:PersonGivenName without reference to any documentation. This will be lost with simplified messages.

On the other hand, the simplified message does not seem very much different than the simplifications we now permit for NIEM JSON. For instance, the following JSON fragment is conforming:

"name": {

"givenName": "Rick"

}

so long as the message specification defines a context object that includes:

"name": "nc:PersonName",

"givenName": "nc:PersonGivenName",

"nc": "http://release.niem.gov/niem/niem-core/4.0/"

And so a similar simplification for NIEM XML may be acceptable after all, if the message specification includes proper mappings and documentation.

**A Remaining Complaint**

Developers also sometimes object to "unnecessary nesting" in NIEM data. This applies to both XML and JSON data. The objection is to data like this:

<nc:PersonBirthDate> "nc:PersonBirthDate": {

<nc:Date>1999-01-01</nc:Date> "nc:Date": "1999-01-01"

</nc:PersonBirthDate> }

Developers often want to know, why not this?

<nc:PersonBirthDate>1999-01-01</nc:PersonBirthDate>

"nc:PersonBirthDate": "1999-01-01"

This nesting is part of the NIEM model. It is not an artifact of the way we represent the NIEM model in XML Schema, and so the introduction of CMF does not help in the same way. There may be a way to satisfy this developer complaint, but it will be more complex than generating different schemas from the CMF model.

**Conclusion**

The development of the Common Model Format and tooling opens up new possibilities for NIEM XML schemas. Simplified schemas that validate the existing messages are possible and will satisfy some developer complaints. There seems to be no reason not to provide these. Simplified XML messages satisfy even more developer complaints, and seem to be consistent with our NIEM JSON specification – but the impact of that change is potentially high, and it will take some time to understand all the consequences. It will of course be essential to consider the best way to introduce these changes to the NIEM community, bearing in mind the impact on existing community knowledge and training resources.