How NIEM Uses XML
(part 1)
Objectives Roadmap
This module supports the following course objective:

- Describe what NIEM is.
- Describe what an IEPD is.
- Comprehend artifacts included in an IEPD.
- Develop artifacts included in an IEPD.
- Package an IEPD.
- Understand advanced XML concepts, as required by NIEM.
- Recognize business skills required to successfully participate in an IEPD development project.
Module Objectives

After completing this module, you should be able to:

- Define use of Namespaces in NIEM.
- Identify and describe Naming Conventions.
- Recognize conceptual structure of NIEM.
- Describe Inheritance.
- Identify and describe Associations.
- Recognize the purpose of Roles.
NIEM Namespaces

- NIEM uses namespaces to organize the model:
  - NIEM-Core (nc:)
  - Individual Domains (intel:, em:, j:, etc.)
  - Code Table authorities

- Allows modification, maintenance, and ownership of different parts of the model without affecting other parts.
NIEM Use of URIs

- URIs look like URLs

```xml
xmlns:nc="http://niem.gov/niem/niem-core/2.0"
```

- Host identifier which specifies the authoritative location for NIEM
- The layer in the NIEM architecture
- Sub-location where NIEM is kept at the host
- Version number
ISO 11179 allows elements to retain some semantic meaning even when viewed with neither context nor definition. “Name” versus “PersonName” and “OrganizationName.”
Representation Terms

- Representation terms are from ebXML Core Component Tech Spec:
  - Amount
  - Binary Object
  - Code
  - DateTime
  - Identifier
  - Indicator
  - Measure
  - Numeric
  - Quantity
  - Text

- There are other non-ebXML terms that could be used.
Exercise 10.1: Naming Conventions

• Create an NIEM-conformant name for the concept of whether an organization is unrepentedly evil.
Solution 10.1: Naming Conventions

- OrganizationUnrepentedlyEvilIndicator
• NIEM consists of elements, attributes, and types.
• Elements are declared as being of certain types.
• More specific types are derived from more general types.
• The type of an element determines what other elements it can contain.
• Complex objects (those that contain other objects) ultimately derive from `s:ComplexObjectType`.
• Simple objects, those that contain actual data, ultimately derive from a number of XML schema data types, such as string, number, and date.
Hierarchies: Type vs. Container (1 of 3)

- nc:PersonNameType derived-from s:ComplexObjectType
- nc:Person contains nc:PersonName
  - of type nc:PersonName
    - nc:PersonGivenName
    - nc:PersonMiddleName
    - nc:PersonSurName
    - nc:PersonLivingIndicator
Hierarchies: Type vs. Container

- niem-xsd:string
- nc:TextType
- nc:ProperNameTextType
- nc:PersonNameTextType
- nc:Person
- nc:PersonName
- nc:PersonGivenName
- nc:PersonMiddleName
- nc:PersonSurName
- nc:PersonLivingIndicator

 contiene-derived-from

of type
Hierarchies: Type vs. Container

contains

niem-xsd:boolean

of type

nc:Person

nc:PersonName

nc:PersonLivingIndicator

nc:PersonGivenName

nc:PersonMiddleName

nc:PersonSurName
Hierarchy Overview

Subject contains Type

Object contains Type

of type

contains

contains

contains

of type

of type

contains

contains

General Form

Subject-Type

contains

Property

of type

Object-Type

of type

of type

of type

of type

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contains

contai
Inheritance

• Specific semantic objects are built from more generic objects, in a process called *Specialization*.

• The more specific objects inherit properties from the more generic object, then add properties special to themselves.

• The more specific objects can act as the source for even more specific semantic objects, which will inherit from all levels above it.
Inheritance

• Specialization only used when an object is truly a special version of the more generic object.
  - Not just a specialized “role” that the more generic object plays.
  - Look for permanence, is the special thing always special.
  - Look at circumstances, can the “specialness” go away in different circumstances?
Implementation in XML

- NIEM uses normal XML Schema extension techniques.
- The more generic object is used as a base.
- The more specific object extends the base to add its own special properties.
- The more specific object also inherits properties from the generic object.
Inheritance

- `nc:OrganizationType`
  - contains `nc:OrganizationName`
  - contains `nc:OrganizationIdentification`
  - Special kind of `ip:AirlineType`
  - Special kind of `em:HospitalType`
    - `ip:Aircraft`
    - contains `ip:Aircraft`
    - contains `em:HospitalFacilityStatus`
Inheritance

nc:OrganizationType

contains

Special kind of

nc:OrganizationName

nc:OrganizationIdentification

contains

ip:AirlineType

contains

Special kind of

ip:Aircraft

em:HospitalType

contains

em:HospitalFacilityStatus

nc:OrganizationName

nc:OrganizationIdentification
Contextual Definitions

• Inheritance creates contextual definitions.
• Concepts exist in the NIEM conceptually, but not by name.
• The concept of *HospitalName* is represented by an `nc:OrganizationName` in the context of the `em:Hospital` object.
• *HospitalName* itself does not appear.
<!-- Simplified snippet from em: domain -->
<xsd:complexType name="HospitalType">
  <xsd:complexContent>
    <xsd:extension base="nc:OrganizationType">
      <xsd:sequence>
        <xsd:element ref="em:HospitalBedCapacityStatus"/>
        <xsd:element ref="em:HospitalFacilityStatus"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="Hospital" type="em:HospitalType"/>
<!-- Simplified snippet from nc: domain -->
<xsd:complexType name="OrganizationType">
  <xsd:complexContent>
    <xsd:extension base="s:ComplexObjectType">
      <xsd:sequence>
        <xsd:element ref="nc:OrganizationName"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
Exercise 10.2: Inheritance

- Consider a Commercial Vehicle.
- Find it in NIEM.
- List all the types from which it inherits.
- How many total elements can a Commercial Vehicle element contain?
Solution 10.2: Inheritance

- Commercial Vehicle inherits from:
  - nc:VehicleType
  - nc:ConveyanceType
  - nc:TangibleItemType
  - nc:ItemType
  - s:ComplexObjectType

- Total Elements: 84
Associations

- Associations are used to link together related objects.
- Associations also contain additional properties specific to the Association itself, such as the dates an association began and ended.
- This better reflects reality than enclosing objects inside of other objects.
Inclusion vs. Association

Person

Vehicle

Person

Owns

Vehicle
Provided Associations

- NIEM contains several pre-defined Associations, linking people, documents, organizations, items, locations, and other NIEM objects in various combinations.
- Associations are typically separate from the objects they connect, although not always.
  - An occasional association will contain an object rather than linking to it.
  - In general, it’s better to link to an object than to contain it within an association.
Multiple Associations

Person

Vehicle

My car, too!

Person

Person

Owns

Vehicle
Implementation in XML

- Associations are implemented via References that point to the objects being associated.
- References use `s:id` and `s:ref` to establish the links.
- Associations can be specialized into more specific relationships.
- NIEM References element names end in “Reference.”
Schema Example

<!-- Simplified snippet from nc: namespace -->
<xsd:complexType name="PersonLocationAssociationType">
  <xsd:complexContent>
    <xsd:extension base="nc:AssociationType">
      <xsd:sequence>
        <xsd:element ref="nc:PersonReference"/>
        <xsd:element ref="nc:LocationReference"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="PersonCurrentLocationAssociation" type="nc:PersonLocationAssociationType"/>

<!-- nc:Location defined elsewhere -->
<!-- nc:Person defined elsewhere -->
Instance Example

<nc:Person s:id="id0001">
  <nc:PersonName>
    <nc:PersonFullName>Joe Witness</nc:PersonFullName>
  </nc:PersonName>
</nc:Person>

<nc:Location s:id="id0002">
  <nc:LocationName>Chez NIEM</nc:LocationName>
</nc:Location>

<nc:PersonCurrentLocationAssociation>
  <nc:PersonReference s:ref="id0001"/>
  <nc:LocationReference s:ref="id0002"/>
</nc:PersonCurrentLocationAssociation>
Exercise 10.3: Associations

• Find the best Association for relating a Person to the Location in which they Reside.
Solution 10.3: Associations

• Okay, but not best:
  - nc:PersonLocationAssociation

• Best
  - nc:ResidenceAssociation
    - Sometimes, one of the targets is implied
Roles

- Roles are special associations that link objects to temporary roles that the objects play. In NIEM, they apply to People, Organizations, and Items.
- Tempting to use specialization to implement roles, such as a police officer, a victim, or a witness....but wrong!!
Specialization Fails

• Specialization doesn’t reflect reality:
   A Person is only a police officer, a victim, or a witness for a particular period of time in a particular context.
   A Person can act in multiple roles. A single person can play the role of a police officer, a victim, and a witness, all at the same time in a particular Incident.
Separate Role Objects

- Roles are separate objects. These objects are not People, Organizations, and Items. They are standalone objects that hold properties specific to the Role and point to the object playing that Role.

![Diagram showing roles and person]

- Witness
- Victim
- Person
XML Implementation

- The Role contains pointers to the objects playing the Role, using `s:id` and `s:ref` via 3 types of RoleOf References:
  - RoleOfPersonReference
    - (for example: Victim, Subject)
  - RoleOfOrganizationReference
    - (for example: CriminalOrganization, LienHolder)
  - RoleOfItemReference
    - (for example: Weapon, CrashVehicle)
<!-- Simplified snippet from j: namespace -->
<xsd:complexType name="WitnessType">
  <xsd:complexContent>
    <xsd:extension base="s:ComplexObjectType">
      <xsd:sequence>
        <xsd:element ref="nc:RoleOfPersonReference"/>
        <xsd:element ref="j:WitnessAccountDescriptionText"/>
        <xsd:element ref="j:WitnessWillTestifyIndicator"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="Witness" type="j:WitnessType"/>

<!-- nc:Person defined elsewhere -->
<nc:Person s:id="id0001">
  <nc:PersonName>
    <nc:PersonFullName>Joe Witness</nc:PersonFullName>
  </nc:PersonName>
</nc:Person>

<j:Witness>
  <nc:RoleOfPersonReference s:ref="id0001"/>
  <j:WitnessAccountDescriptionText>
    I saw everything!
  </j:WitnessAccountDescriptionText>
  <j:WitnessWillTestifyIndicator>True</j:WitnessWillTestifyIndicator>
</j:Witness>
Exercise 10.4: Roles

• Given this simplified snippet of Schema from the j: namespace, add a Victim role to the previous instance example:

```xml
<xsd:complexType name="VictimType">
   <xsd:complexContent>
      <xsd:extension base="s:ComplexObjectType">
         <xsd:sequence>
            <xsd:element ref="nc:RoleOfPersonReference"/>
            <xsd:element ref="nc:RoleOfOrganizationReference"/>
            <xsd:element ref="j:VictimSeeksProsecutionIndicator"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>

<xsd:element name="Victim" type="j:VictimType"/>
```
Solution 10.4: Roles

<nc:Person s:id="id0001">
  <nc:PersonName>
    <nc:PersonFullName>Joe Witness</nc:PersonFullName>
  </nc:PersonName>
</Person>

<j:Witness>
  <nc:RoleOfPersonReference s:ref="id0001"/>
  <j:WitnessWillTestifyIndicator>
    True
  </j:WitnessWillTestifyIndicator>
</j:Witness>

<j:Victim>
  <nc:RoleOfPersonReference s:ref="id0001"/>
  <j:VictimSeeksProsecutionIndicator>
    True
  </j:VictimSeeksProsecutionIndicator>
</j:Victim>
Module Summary

After completing this module, you should be able to:

- Define use of Namespaces in NIEM.
- Identify and describe Naming Conventions.
- Recognize conceptual structure of NIEM.
- Describe Inheritance.
- Identify and describe Associations.
- Recognize the purpose of Roles.
Creative Commons

Attribution-ShareAlike 2.0
You are free to
• Copy, distribute, display, and perform the work
• Make derivative works
• Make commercial use of the work

Under the following conditions
• For any reuse or distribution, you must make clear to others the license terms of this work
• Any of these conditions can be waived, if you get permission from the copyright holder

Your fair use and other rights are in no way affected by the above
This is a human-readable summary of the Legal Code (the full license) and Disclaimer

This page is available in the following languages
Català, Deutsch, English, Castellano, Suomeksi, français, hrvatski, Italiano, 日本語, Nederlands, Português, and 中文（繁）

Learn how to distribute your work using this license