WHITE**PAPER**

APPLICATIONS SOFTWARE TECHNOLOGY LLC

Building an IEPD for NIEM Data Model

Roshan Parab

April 2016

CONTENTS

1 DEFINITIONS, ABBREVIATIONS, AND ACRONYMS

The table below provides definitions for the acronyms used in this document:

Acronym	Acronym Description
NIEM	National Information Exchange Model
GJXDM	Global Justice XML Data Model
IEP	Information Exchange Package
IEPD	Information Exchange Package Documentation
SSGT	Schema Subset Generation Tool

2 ABSTRACT

The National Information Exchange Model (NIEM) is a U.S. government-sponsored initiative to facilitate information sharing among public and private sector organizations. Over the years, NIEM has matured and its data model is widely used. It has become the de-facto standard for information exchange across all levels of government and private industry.

With NIEM, there is an option to use an existing data model or to develop a custom model suited specifically for your organization. Implementation of NIEM for a specific data exchange is known as IEPD (Information Exchange Package Documentation). Developing an IEPD is a lengthy and complex task, as documentation is not readily available and there are few examples of practical, end-to-end scenarios. There are a few tools available that aid in building an IEPD, each with its own features. This paper explains the process of creating an IEPD from scratch, and details the tools and methodologies used at each stage.

3 INTRODUCTION

This paper outlines the approach for using the NIEM data model to create new IEPDs, which facilitate information exchange.

4 What is NIEM

NIEM is a XML-based information exchange framework. It is a community-driven, government-wide, standards-based approach to exchanging information. NIEM represents a collaborative partnership of agencies and organizations across all levels of government (federal, state, tribal, and local) and within the private industry. The purpose of this partnership is to effectively and efficiently share critical information at key decision points throughout justice, public safety, emergency, disaster management, intelligence, and homeland security enterprises.

NIEM is an outgrowth of the United States Department of Justice's Global Justice XML Data Model (GJXDM) project. It is now being expanded to include other federal and state agencies, such as the Office of the Director of National Intelligence, United States Department of Defense, and the Federal Bureau of Investigation. In addition, NIEM is expanding into multiple states, including but not limited to Texas, Florida, New York, and Pennsylvania.

5 MODELING NIEM EXCHANGE

Information Exchange Package (IEP)

In NIEM, an information exchange, also known as an IEP, is a data set that is exchanged between two entities (e.g., systems, agencies, departments, etc.) for a specific business purpose. In simpler terms, it is a NIEM-conformant XML that is exchanged between two machines.

Information Exchange Package Documentation (IEPD)

IEPD is a set of artifacts that define the content and structure of the IEP. The IEPD includes information regarding the IEP, such as documentation, sample XML instances, business rules, and more. An IEPD is the final product of the NIEM information exchange development process, also known as the IEPD Lifecycle.

6 Solution Details

This section outlines the steps to build an IEPD from scratch.

Identify Existing IEPDs

The first step in developing information exchange is to identify existing exchanges, which may be similar to the information exchange being developed.

To search for existing IEPDs, NIEM provides an IEPD Clearing House: a database of IEPDs submitted by individuals and organizations who have implemented GJXDM and NIEM.

IEPD Clearing Houses can be located here:

- http://iepd.custhelp.com/
- https://tools.niem.gov/niemtools/iepdt/search/index.iepd

In addition to the above, it is also advisable to leverage the NIEM community during the development process, which can be found here: https://www.niem.gov/communities/Pages/communities.aspx

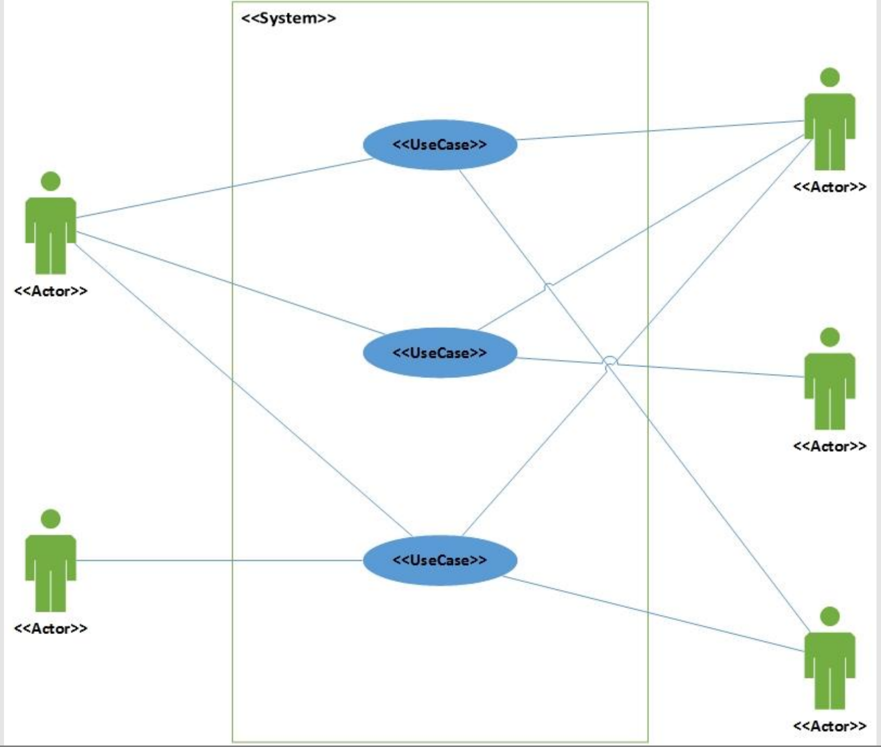
There are currently 207 IEPDs available, including US Department of Justice reference IEPD for LEXS 3.1 using NIEM 2.0 model. Once it has been confirmed that none of the existing IEPD fulfill the need at hand, proceed with the next step for creation of IEPD.

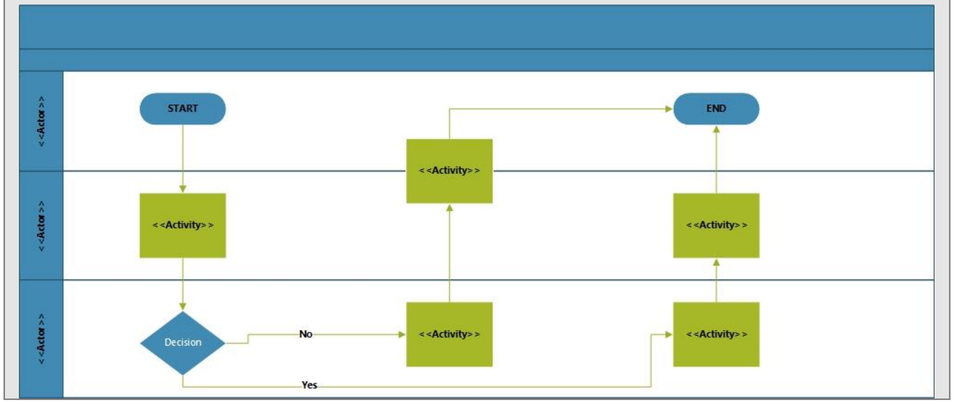
Modeling the Information Exchange

Modeling creates a graphical representation of the information exchange requirements and helps build the information exchange.

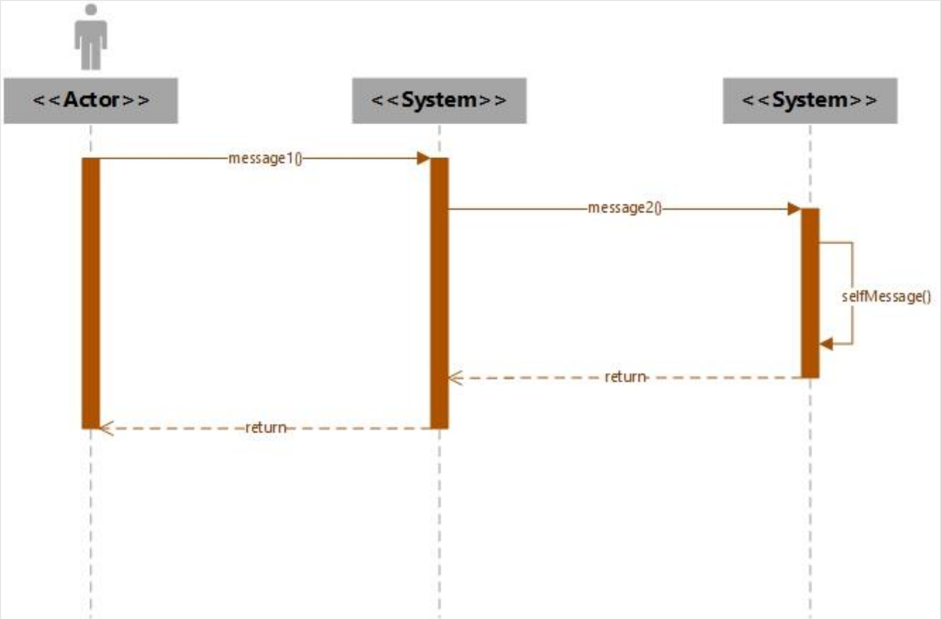
It is advisable to include the following diagrams in an IEPD:

Use Case Diagrams

Purpose	Functionality
Description	Describes system interactions by the various actors, as well as the system functionality.
Depiction of a Use-Case Diagram  	

Business process diagrams	
Purpose	Activity Sequencing
Description	Describes activity sequencing (like a workflow diagram) with stakeholders, activities, decisions, flow, and triggering events or results.
Depiction of Business Process Diagram  <<Actor>> <<Actor>> <<Actor>> START END <<Activity>> <<Activity>> <<Activity>> <<Activity>> <<Activity>> Decision No Yes	

Sequence diagrams	
Purpose	Message Flow
Description	Describes how applications or systems interact with one another and the messages sent between them.

Depiction of a Sequence Diagram
 <<Actor>>
<<System>>
<<System>>
message1()
message2()
selfMessage()
return
return	

Define the Data Model and Exchange Requirements

We begin by defining the data model for the information exchange, known as ECM. The ECM should describe:

a. The objects,

b. The elements within the objects,

c. Associations between those objects, and

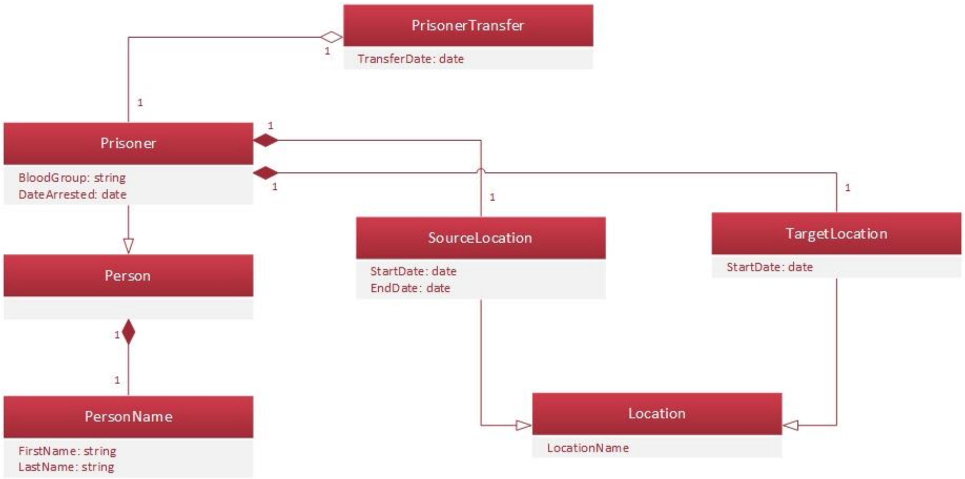
d. Business rules and conditions (if any).

Capture the information in a spreadsheet, such as the one below.

Source Container Type (Object)	Complex, reusable data structures where related elements are grouped together. Examples include Person (with elements First Name, Last Name, SSN, and Phone) and Location (with elements Street Address, City, State, and Zip Code).
Source Element	This is the name of the field / tag. E.g., First Name, Last Name, Phone Number, etc.
Data Type	The structural representation or format of the element. E.g., string, date, number, Boolean, state code set, person data type, location data type, etc.
Definition	Descriptive meaning of elements.
Cardinality	The minimum and maximum number of times an element may appear in the instance. E.g., Last Name may occur once only, Phone Number may occur zero to many times.
Code Sets	Lists of allowable values. They may come from a standard list or may be custom-defined. E.g., State code list, Hair color list, etc.
Conditions / Business Rules	Restrictions on the content that need to be represented. E.g., Restrictions on individual values (such as a number that cannot be less than zero or a string that must have a certain number of characters) Conditional restraints (such as if the SSN is not provided, then Last Name and Birth Date are required), etc.

Let's examine an example where a prisoner transfer must occur. Assume the rule is such that during the tenure of imprisonment, a person cannot be kept in same jail for more than x years. In such cases, he must be transferred to a jail in another state. This demands transfer of prisoner data from one police department to another.

The data model can be represented using Unified Modelling Language (UML) class diagrams. The UML diagram for the scenario listed above would appear as follows:



The following list contains attributes that are relevant to defining the ECM:

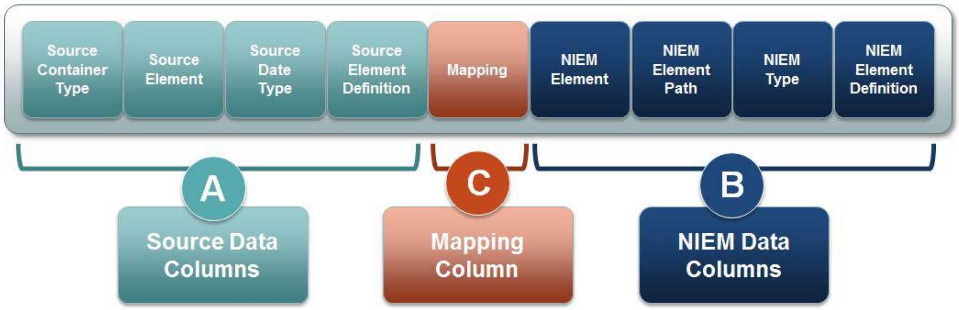
Source Container Type	Source Element	Data Type	Source Element Definition	Cardinality	Business Rules
PersonName	First Name	String	First name of a person	0..1	
PersonName	Last Name	String	Last name of a person	0..1	
Person	Blood Group	String	Blood group of a person	0..1	Blood Group can be either of A, B, O types
Prisoner	Date Arrested	Date	Date person was arrested	1..1	
Location	City	String	Name of City	0..1	
Address	Country	String	Name of the country	0..1	

* This example captures minimal information required to explain the concept.

Note – When defining an element, if a corresponding NIEM element is already known at this stage, then the NIEM values may be used in place of local ones for element name, data type, and definition.

Define the Mapping Document

Map the elements and objects identified in the Exchange Content Model to the elements from the NIEM model.

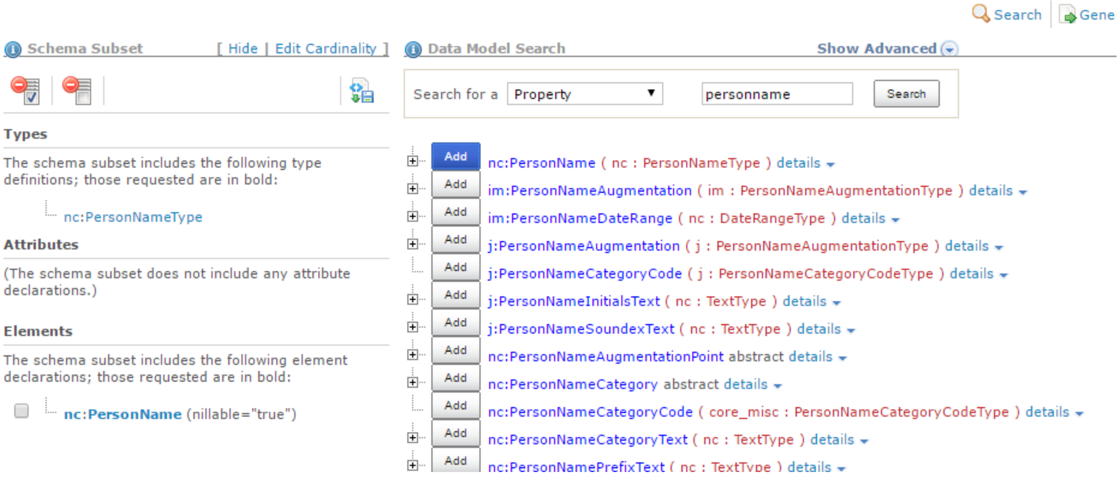


You can extend the spreadsheet template used above to achieve the same.

Source Container Type	Source Element	Data Type	Mapping	NIEM Element	NIEM Element Path	NIEM Type	NIEM Element Definition
Person	First Name	String	Match found	nc:PersonGivenName	PersonName/nc:PersonGivenName	nc:PersonNameTextType	A first name of a person
Person	Last Name	String	Match found	nc:PersonGivenName	nc:PersonName/nc:PersonSurname	nc:PersonNameTextType	A last name of a person
Person	Blood Group	String	No match found				

To find the matching NIEM components use the Schema Subset Generation Tool (SSGT, https://tools.niem.gov/niemtools/ssgt/index.iepd).

Search for "personname" and click on add to add "nc:PersonName" element



Search for a Property ▼ personname Search

Types

The schema subset includes the following type definitions; those requested are in bold:

- nc:PersonNameType

Attributes

(The schema subset does not include any attribute declarations.)

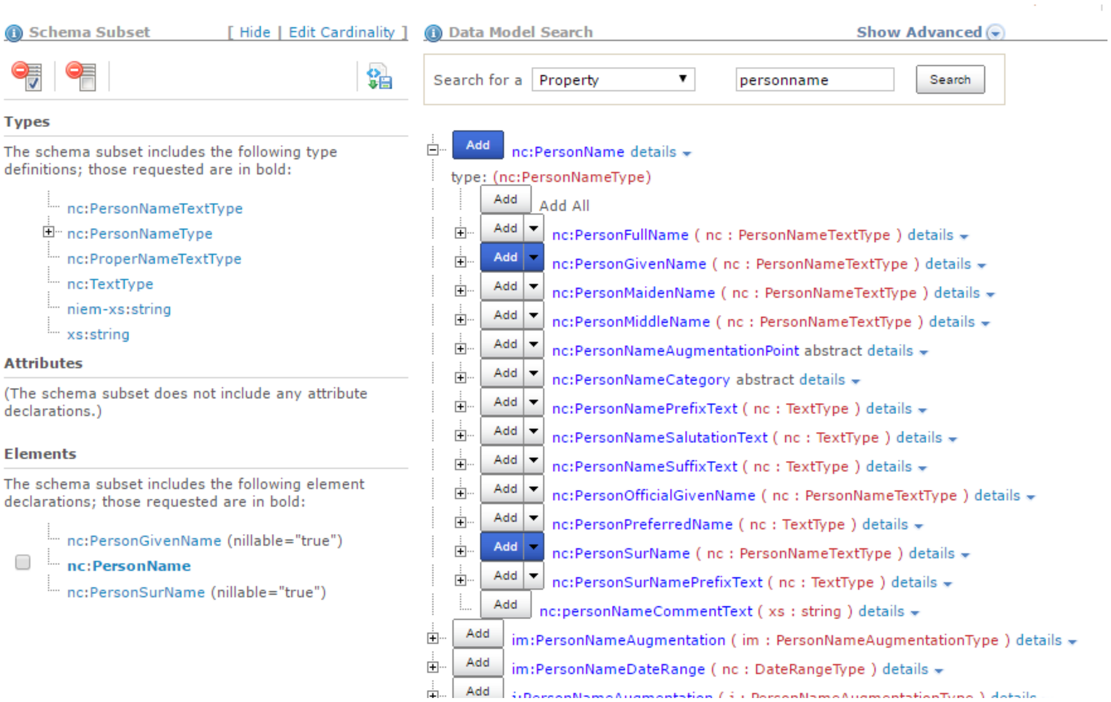
Elements

The schema subset includes the following element declarations; those requested are in bold:

□ **nc:PersonName** (nillable="true")

- Add nc:PersonName (nc : PersonNameType) details
- Add im:PersonNameAugmentation (im : PersonNameAugmentationType) details
- Add im:PersonNameDateRange (nc : DateRangeType) details
- Add j:PersonNameAugmentation (j : PersonNameAugmentationType) details
- Add j:PersonNameCategoryCode (j : PersonNameCategoryCodeType) details
- Add j:PersonNameInitialsText (nc : TextType) details
- Add j:PersonNameSoundexText (nc : TextType) details
- Add nc:PersonNameAugmentationPoint abstract details
- Add nc:PersonNameCategory abstract details
- Add nc:PersonNameCategoryCode (core_misc : PersonNameCategoryCodeType) details
- Add nc:PersonNameCategoryText (nc : TextType) details
- Add nc:PersonNamePrefixText (nc : TextType) details

If nc:PersonName is added, it does not mean its sub-elements are added as well. To add those, expand nc:PersonName and add the appropriate one.



Search for a Property ▼ personname Search

Types

The schema subset includes the following type definitions; those requested are in bold:

- nc:PersonNameTextType
- nc:PersonNameType
- nc:ProperNameTextType
- nc:TextType
- niem-xs:string
- xs:string

Attributes

(The schema subset does not include any attribute declarations.)

Elements

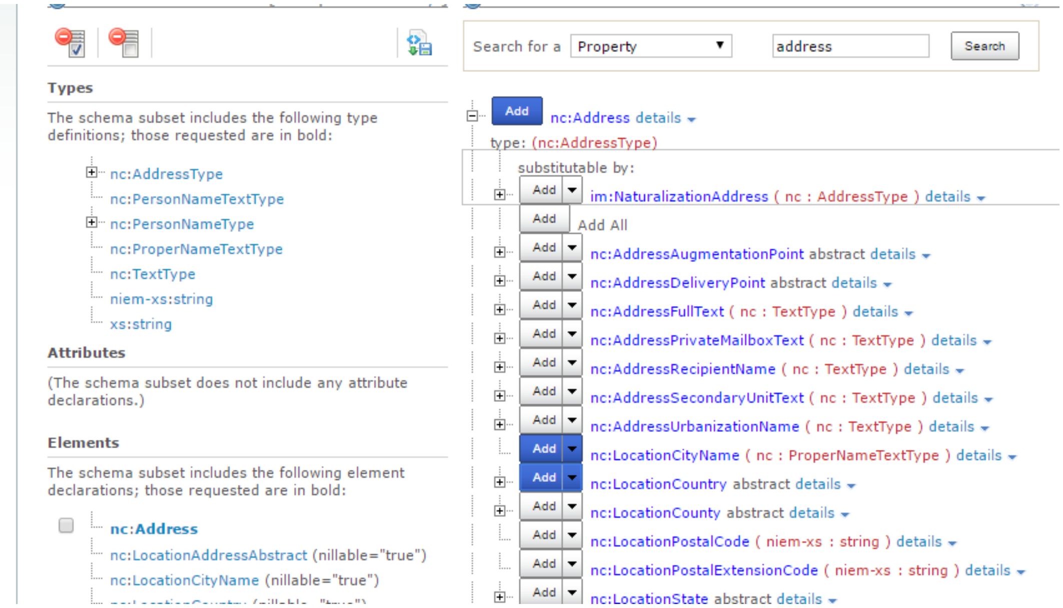
The schema subset includes the following element declarations; those requested are in bold:

- nc:PersonGivenName (nillable="true")
- □ **nc:PersonName**
- nc:PersonSurName (nillable="true")

- Add nc:PersonName details
  - type: (nc:PersonNameType)
  - Add Add All
  - Add nc:PersonFullName (nc : PersonNameTextType) details
  - Add nc:PersonGivenName (nc : PersonNameTextType) details
  - Add nc:PersonMaidenName (nc : PersonNameTextType) details
  - Add nc:PersonMiddleName (nc : PersonNameTextType) details
  - Add nc:PersonNameAugmentationPoint abstract details
  - Add nc:PersonNameCategory abstract details
  - Add nc:PersonNamePrefixText (nc : TextType) details
  - Add nc:PersonNameSalutationText (nc : TextType) details
  - Add nc:PersonNameSuffixText (nc : TextType) details
  - Add nc:PersonOfficialGivenName (nc : PersonNameTextType) details
  - Add nc:PersonPreferredName (nc : TextType) details
  - Add nc:PersonSurName (nc : PersonNameTextType) details
  - Add nc:PersonSurNamePrefixText (nc : TextType) details
  - Add nc:personNameCommentText (xs : string) details
- Add im:PersonNameAugmentation (im : PersonNameAugmentationType) details
- Add im:PersonNameDateRange (nc : DateRangeType) details
- Add j:PersonNameAugmentation (j : PersonNameAugmentationType) details

One can view the added elements on the left hand side in the above screenshot.

Search for “address” and click on “add” to add “nc:Address” element.



Types

The schema subset includes the following type definitions; those requested are in bold:

- nc:AddressType
- nc:PersonNameTextType
- nc:PersonNameType
- nc:ProperNameTextType
- nc:TextType
- niem-xs:string
- xs:string

Attributes

(The schema subset does not include any attribute declarations.)

Elements

The schema subset includes the following element declarations; those requested are in bold:

- **nc:Address**
- nc:LocationAddressAbstract (nillable="true")
- nc:LocationCityName (nillable="true")

Search for a Property ▾ address Search

Add nc:Address details ▾

type: (nc:AddressType)

substitutable by:

- Add ▾ im:NaturalizationAddress (nc : AddressType) details ▾

Add Add All

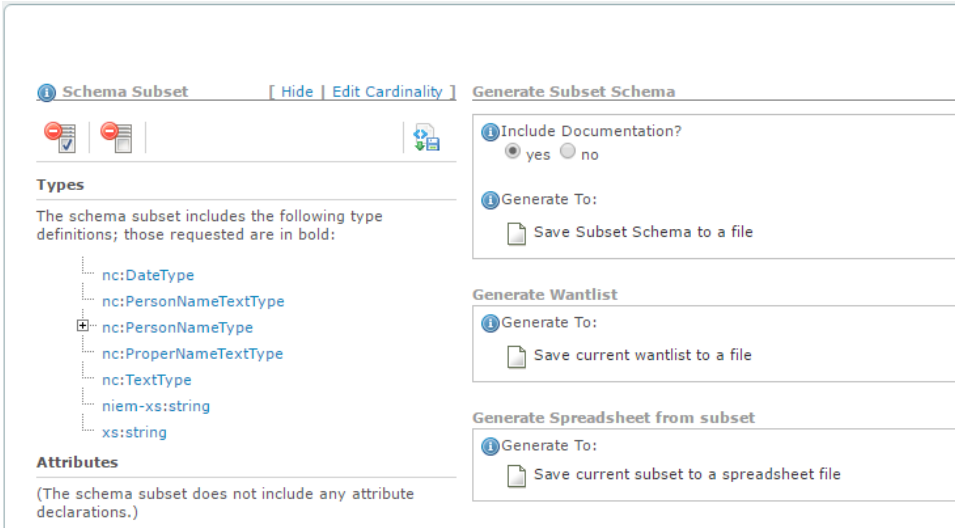
- Add ▾ nc:AddressAugmentationPoint abstract details ▾
- Add ▾ nc:AddressDeliveryPoint abstract details ▾
- Add ▾ nc:AddressFullText (nc : TextType) details ▾
- Add ▾ nc:AddressPrivateMailboxText (nc : TextType) details ▾
- Add ▾ nc:AddressRecipientName (nc : TextType) details ▾
- Add ▾ nc:AddressSecondaryUnitText (nc : TextType) details ▾
- Add ▾ nc:AddressUrbanizationName (nc : TextType) details ▾
- Add ▾ nc:LocationCityName (nc : ProperNameTextType) details ▾
- Add ▾ nc:LocationCountry abstract details ▾
- Add ▾ nc:LocationCounty abstract details ▾
- Add ▾ nc:LocationPostalCode (niem-xs : string) details ▾
- Add ▾ nc:LocationPostalExtensionCode (niem-xs : string) details ▾
- Add ▾ nc:LocationState abstract details ▾

Build and Validate NIEM-Conformant Schemas

The purpose of this phase is to create the XML documents that define the information exchange for the IEPD.

1. Subset Schema

   - This is the XML Schema that defines the subset of the NIEM model used in this exchange. It is the set of local components that map to NIEM model.
   - As described in previous steps, once matching NIEM components are found (using SSGT), they can be exported to form subset schema which will be later packaged in an IEPD and referenced in Exchange schema.
   - Save the subset to the base-xsd subfolder in the IEPD package.
   - Also save the subset wantlist (save file) for future purpose.
   - The Elements captured in the wantlist represent the elements that are going to be part of the subset schema.
   - The subset schema is exported as ZIP archive and the wantlist as a XML file.



Types

The schema subset includes the following type definitions; those requested are in bold:

- nc:DateType
- nc:PersonNameTextType
- nc:PersonNameType
- nc:ProperNameTextType
- nc:TextType
- niem-xs:string
- xs:string

Attributes

(The schema subset does not include any attribute declarations.)

Include Documentation?
● yes ○ no

Generate To:
Save Subset Schema to a file

Generate Wantlist

Generate To:
Save current wantlist to a file

Generate Spreadsheet from subset

Generate To:
Save current subset to a spreadsheet file

```
PrisonerExchange
          |____base-xsd
                   |________niem
                   |________wantlist.xml
                   |________xml-catalog.xml
```

2. Extension Schema

   For the set of Types, Elements, and Attributes not available within NIEM but needed for the exchange, we create a separate schema for them that are defined as extension schemas.

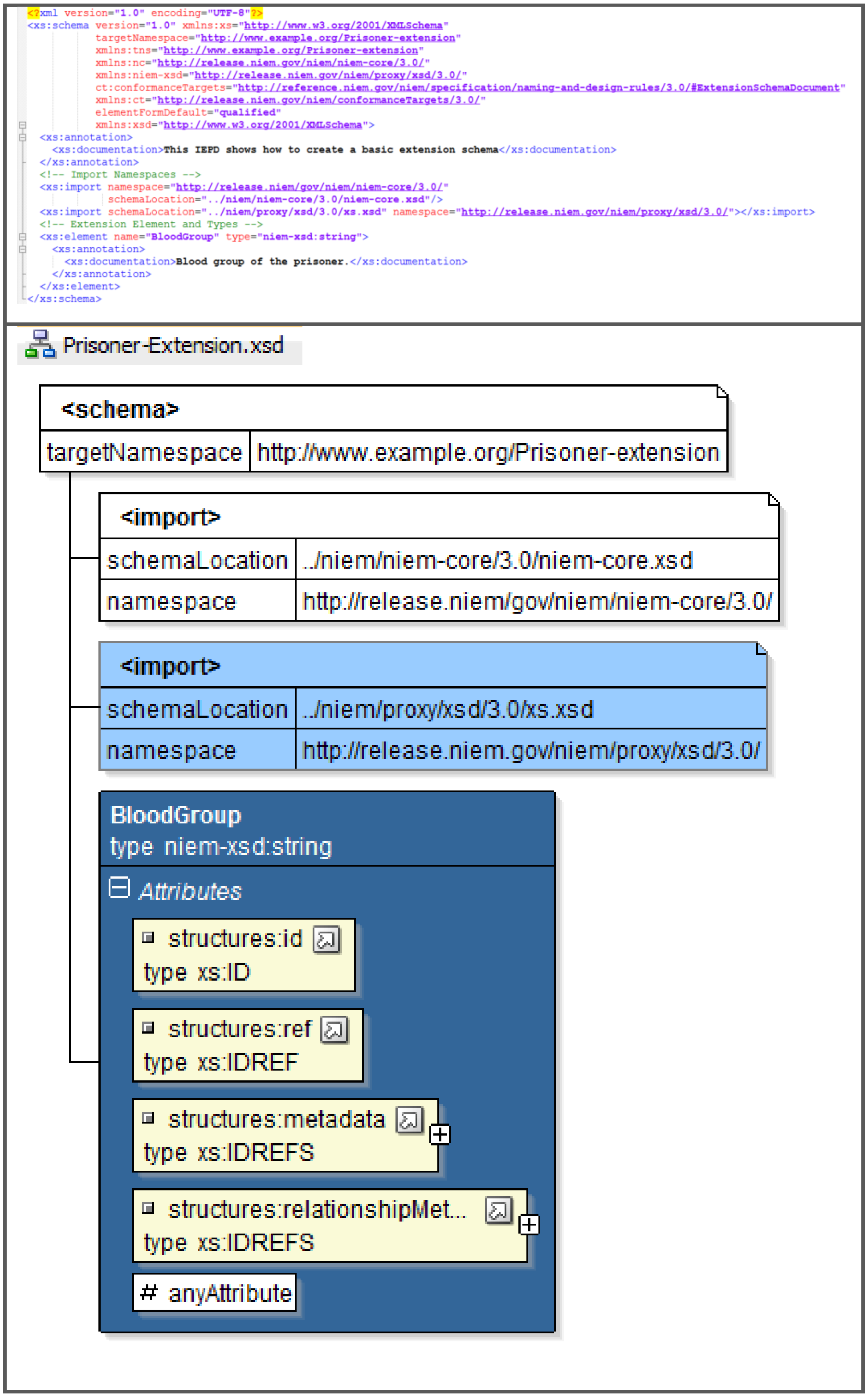
   Though extension schemas are not a mandatory artifact of the IEPD, most IEPDs will have one, as NIEM may not include all of the elements needed for a given information exchange.

   When defining an extension schema, there should be:

   - An .xsd file with its own namespace;
   - References and prefixes to NIEM and W3C namespaces; and
   - Import statements for any referenced schemas.

Below is an example where the element name BloodGroup, which was not available in NIEM, was then defined as an extension to NIEM by creating an extension schema.

Prisoner-Extension.xsd



```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://www.example.org/Prisoner-extension"
           xmlns:tns="http://www.example.org/Prisoner-extension"
           xmlns:nc="http://release.niem.gov/niem/niem-core/3.0/"
           xmlns:niem-xsd="http://release.niem.gov/niem/proxy/xsd/3.0/"
           ct:conformanceTargets="http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/#ExtensionSchemaDocument"
           xmlns:ct="http://release.niem.gov/niem/conformanceTargets/3.0/"
           elementFormDefault="qualified"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xs:annotation>
    <xs:documentation>This IEPD shows how to create a basic extension schema</xs:documentation>
  </xs:annotation>
  <!-- Import Namespaces -->
  <xs:import namespace="http://release.niem/gov/niem/niem-core/3.0/"
             schemaLocation="../niem/niem-core/3.0/niem-core.xsd"/>
  <xs:import schemaLocation="../niem/proxy/xsd/3.0/xs.xsd" namespace="http://release.niem.gov/niem/proxy/xsd/3.0/"></xs:import>
  <!-- Extension Element and Types -->
  <xs:element name="BloodGroup" type="niem-xsd:string">
    <xs:annotation>
      <xs:documentation>Blood group of the prisoner.</xs:documentation>
    </xs:annotation>
  </xs:element>
</xs:schema>
```

Prisoner-Extension.xsd

<schema>	
targetNamespace	http://www.example.org/Prisoner-extension

<import>	
schemaLocation	../niem/niem-core/3.0/niem-core.xsd
namespace	http://release.niem/gov/niem/niem-core/3.0/

<import>	
schemaLocation	../niem/proxy/xsd/3.0/xs.xsd
namespace	http://release.niem.gov/niem/proxy/xsd/3.0/

BloodGroup
type niem-xsd:string

Attributes

structures:id
type xs:ID

structures:ref
type xs:IDREF

structures:metadata
type xs:IDREFS

structures:relationshipMet...
type xs:IDREFS

anyAttribute

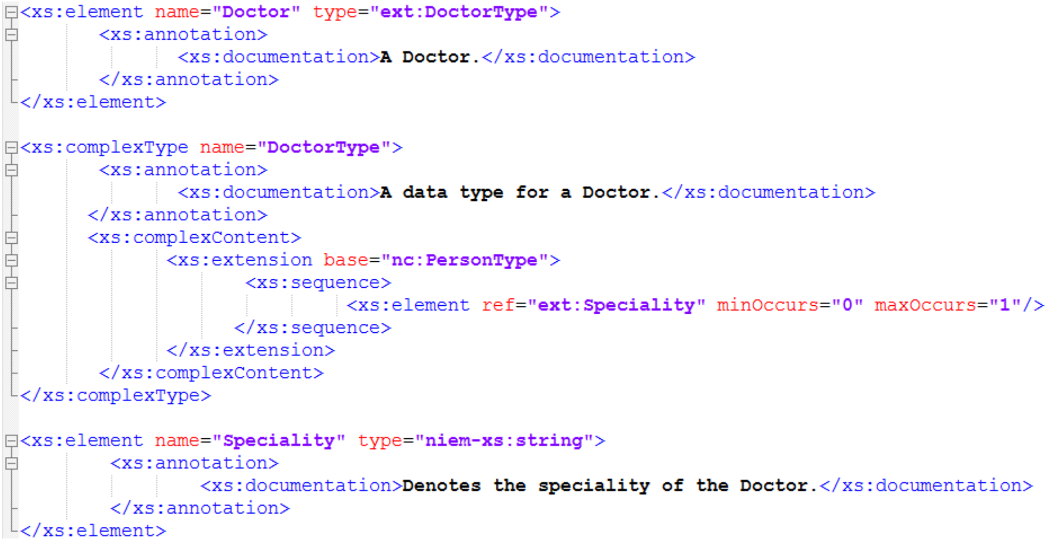
Extension schemas are stored in the base-xsd subfolder in the IEPD package.

Example:

```
PrisonerExchange
         |____base-xsd
                 |________extension
                        |________ Prisoner-Extension.xsd
```

Often, we come across scenarios in which we need to define new complex types in the extension schema.

For example, there is no equivalent of Doctor in the NIEM data model. However, we know that Doctor is a title for a person and can be defined as an extension of PersonType (nc:PersonType), which is already defined in NIEM model.



```
<xs:element name="Doctor" type="ext:DoctorType">
       <xs:annotation>
              <xs:documentation>A Doctor.</xs:documentation>
       </xs:annotation>
</xs:element>

<xs:complexType name="DoctorType">
       <xs:annotation>
              <xs:documentation>A data type for a Doctor.</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
             <xs:extension base="nc:PersonType">
                    <xs:sequence>
                             <xs:element ref="ext:Speciality" minOccurs="0" maxOccurs="1"/>
                   </xs:sequence>
             </xs:extension>
       </xs:complexContent>
</xs:complexType>

<xs:element name="Speciality" type="niem-xs:string">
        <xs:annotation>
                <xs:documentation>Denotes the speciality of the Doctor.</xs:documentation>
        </xs:annotation>
</xs:element>
```

If the new type is not based on an existing type in NIEM, the type must be based on s:ObjectType, which is the root of all complex types in NIEM and is defined in the namespace.

Depending on the complexity of the IEPD, we can have single or multiple extension schemas defined.

3. Exchange Schema

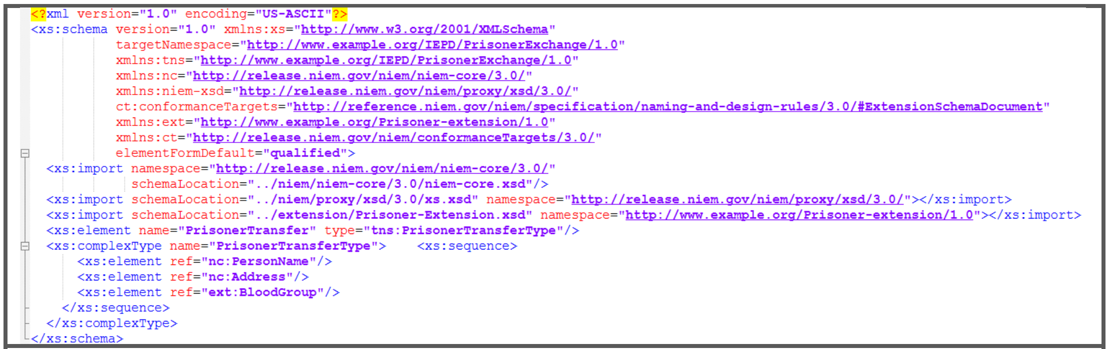
   Exchange schema defines the root element for the information exchange. Exchange schemas tie all of the other schemas together. They import, reuse, and arrange data from other schemas. They are mandatory as per IEPD specification.

   Points to note:

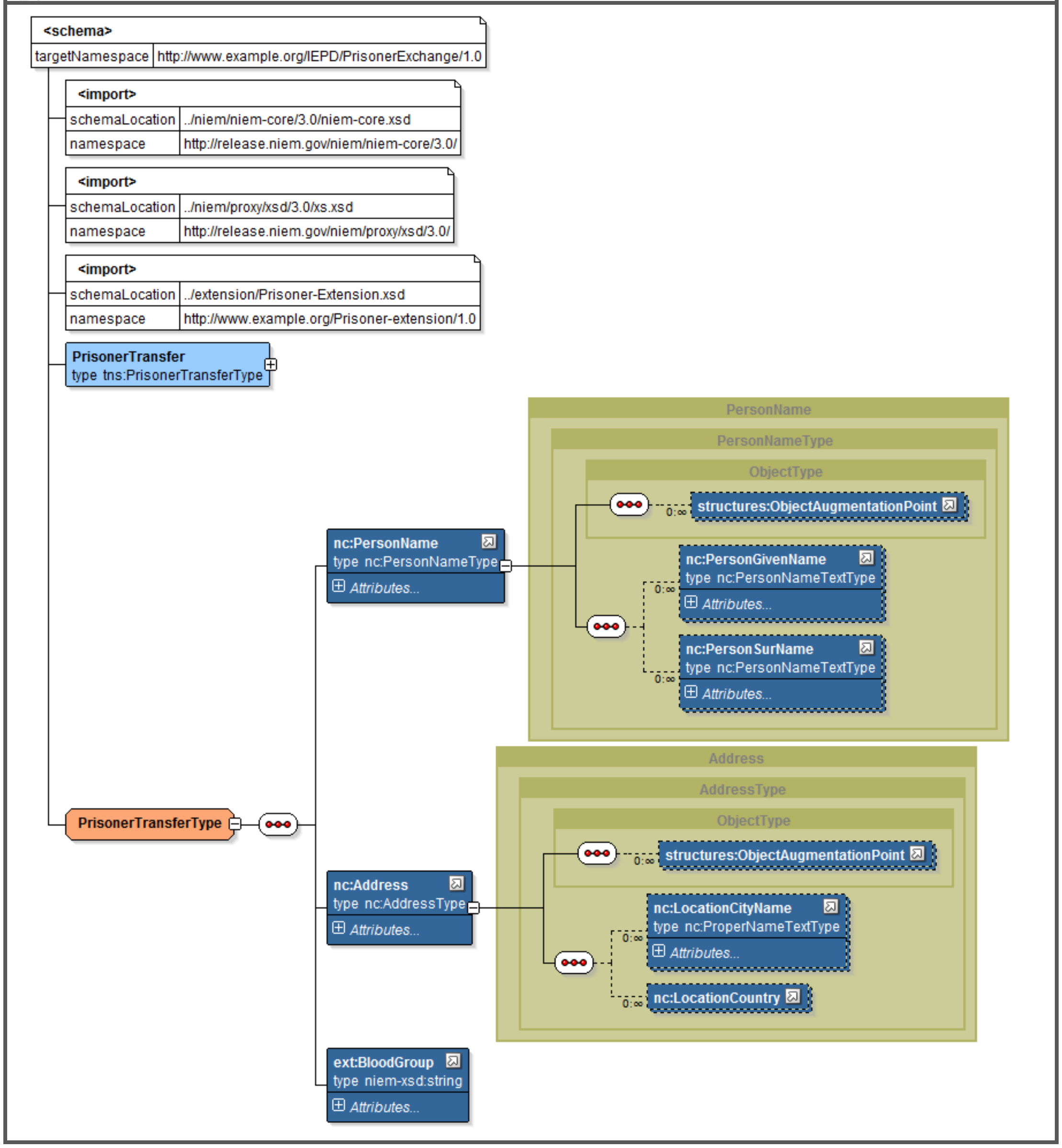
- An IEPD can have multiple exchange schemas since it is possible that multiple exchanges can be documented through a single IEPD.
- They are unique to every exchange and each exchange must use a separate namespace.

Below is the PrisonerTransfer example, which we used above:

PrisonerTransfer-Exchange.xsd



```
<?xml version="1.0" encoding="US-ASCII"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://www.example.org/IEPD/PrisonerExchange/1.0"
           xmlns:tns="http://www.example.org/IEPD/PrisonerExchange/1.0"
           xmlns:nc="http://release.niem.gov/niem/niem-core/3.0/"
           xmlns:niem-xsd="http://release.niem.gov/niem/proxy/xsd/3.0/"
           ct:conformanceTargets="http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/#ExtensionSchemaDocument"
           xmlns:ext="http://www.example.org/Prisoner-extension/1.0"
           xmlns:ct="http://release.niem.gov/niem/conformanceTargets/3.0/"
           elementFormDefault="qualified">
  <xs:import namespace="http://release.niem.gov/niem/niem-core/3.0/"
             schemaLocation="../niem/niem-core/3.0/niem-core.xsd"/>
  <xs:import schemaLocation="../niem/proxy/xsd/3.0/xs.xsd" namespace="http://release.niem.gov/niem/proxy/xsd/3.0/"></xs:import>
  <xs:import schemaLocation="../extension/Prisoner-Extension.xsd" namespace="http://www.example.org/Prisoner-extension/1.0"></xs:import>
  <xs:element name="PrisonerTransfer" type="tns:PrisonerTransferType"/>
  <xs:complexType name="PrisonerTransferType">    <xs:sequence>
      <xs:element ref="nc:PersonName"/>
      <xs:element ref="nc:Address"/>
      <xs:element ref="ext:BloodGroup"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```



Once the schemas have been designed, it is recommended to have them verified as per NIEM standards. NIEM provides online support for conformance validation, which can be accessed here:

https://tools.niem.gov/contesaNIEM/

Assemble Artifacts and Document the IEPD

The purpose of this phase is to complete the IEPD documentation and then assemble all the artifacts in a single, self-contained, portable archive file (e.g., a ZIP file).

Artifact Name	Description	Notes
MPD-Catalog (Mandatory)	XML document that contains metadata about the IEPD.	• Name must be "mpd-catalog.xml" • Structure must conform to the mpd-catalog-3.0.xsd • Must be placed in the root directory of the IEPD Sample mpd-catalog.xml
ReadMe Artifact (Mandatory)	Describes the information exchange. This artifact should briefly describe the IEPD. • Purpose Scope • Business value • Exchange information • Senders/Receivers, interactions • References to other documentation	This artifact name must appear in the MPD catalog.
Change Log (Mandatory)	This artifact describes the changes applied to the IEPD since its previous version. If this is the first version, then the change log simply records its release date.	This artifact name must appear in the MPD catalog.
NIEM Schema Subset (Mandatory)	Output from SGST from previous stage.	
Extension Schema Document(s)	The locally defined schema(s) from the previous stage.	
External Schema Document(s)	Any external standards reused from the previous stage.	
XML Catalogs (Mandatory)	An XML instance that describes mappings between external schema references and locally-cached equivalents.	
Conformance Assertion	An artifact that provides a declaration that the IEPD conforms to relevant NIEM specifications and associated rules, including NIEM Conformance 3.0, NIEM Naming and Design	

	Rules 3.0, NIEM Conformance Targets Attribute Specification 3.0, and NIEM MPD Specification 3.0.	
NIEM Subset Wantlist	A save file generated by the SSGT so that a subset may be reloaded and modified at a future date.	
Business Rules	Formal or informal statements that describe business policy or procedure, to define or constrain some aspect of a process or procedure.	

7 Business Benefits

The benefits of NIEM are well known, hence the NIEM model; its methodology have been widely implemented for data exchange across government and private players in the United States.

This paper should have assisted you in reducing the time to develop your IEPD, removing the need to dig through lengthy documentation.

8 SUMMARY

This document provided the following:

- A high-level introduction to NIEM;
- Terminologies used for NIEM and modeling its exchange;
- A detailed explanation of the lifecycle of IEPD development, with illustrative diagrams at each stage; and
- An overview of the effort involved in developing an IEPD from scratch.

About the Author:

Roshan Parab

Mr. Parab is an Oracle Solutions Architect with over nine years of experience. He is specialized in application- and enterprise-based integration, using Oracle Fusion Middleware and Java/J2EE-based applications. Mr. Parab has worked on large- and medium-scale projects involving Oracle Suite, Oracle Service Bus, Oracle Data Integrator, Oracle Database, and Java, helping to align customer applications with SOA objectives. He has worked for clients across the United States and Europe in various industries, including Media & Entertainment, Energy, and Insurance & Pharmaceuticals. Past customers include: 20th Century Fox, First American, and Pfizer Pharmaceuticals.

Building an IEPD for NIEM Data Model
April 2016

Specialized. Recognized. Preferred. The right partner makes all the difference.

AST LLC is a privately held company founded in 1995 to serve commercial and public sector organizations in utilizing the full potential of their investment in Oracle Applications. We have built our business by providing top quality full life cycle and turn-key consulting services for Oracle implementation and upgrade projects, among other specific services.

Our mission is to ensure that you, as a valued client, receive the highest level of expertise and personalized service for your Enterprise Applications. We endeavor to help you realize the full benefit from your investment in software applications and integrate them seamlessly to your business processes.

AST LLC
1755 Park Street, Suite 100
Naperville, Illinois 60563

Phone: 888-278-0002
Fax: 630-778-1179
www.astcorporation.com