Open Source NIEM UML Tool Tutorial

Basic Information Exchange

Contents

Purpose, Overview, Background

Welcome to the NIEM-UML Basic Exchange Tutorial on the Open Source NIEM UML Modeling Tool, a tool to design and generate a NIEM Conformant MPD/IEPD to support your information exchanges. This tutorial will present the installation and navigation of the NIEM-UML tool and provide instruction on how to model a basic exchange with the NIEM profile. NIEM Stereotypes to be covered include basic object types, association types, property holders types, and role types, through the modeling of a NIEM Core Subset schema, an extension schema, and an exchange schema.

An understanding of NIEM, XML, and UML along with some background in information sharing exchanges is recommended as pre-requisites to this tutorial. The NIEM Program Management Office has published several webinars that can be found at https://www.niem.gov/training/Pages/webinars.aspx . If you are already familiar with the NIEM Schema Subset Generator Tool (SSGT), you will find the NIEM-UML Tool to be complementary to the SSGT experience.

UML (Unified Modeling Language) is an OMG specification and arguably, the most widely adopted standard for application structure, behavior, and architecture, as well as business requirements and data structure. It supports Model-Driven Architecture (MDA) and helps unify application design and development. In September, 2013, the OMG officially finalized a NIEM Profile for UML, NIEM-UML 2.1. This profile generates 100% NIEM-conformant information exchanges and provides a visual representation of those exchanges that is understandable to both technical and business users.

In summary, NIEM-UML is a new NIEM specification that provides for modeling NIEM in UML and producing or reverse engineering information exchange technical specifications using Model Driven Architecture. This reduces the time, cost, and learning curve of information exchange using NIEM. MDA also provides for other aspects of the information sharing solution, such as: business processes, SOA services, and back-end system integration. Since NIEM-UML generates 100% NIEM conformant technical specifications, NIEM-UML Architects and Developers don't need to worry about as much about the technology details. NIEM-UML can be extended to support other technologies, such as JSON and the semantic web.

Development Approach

Through PM-ISE funding, SEARCH built a NIEM-UML tool through extending available open source platforms with the NIEM profile. For the tool, SEARCH chose the Eclipse Software Development Kit as the development platform, GITHUB as the development environment, and the Papyrus Component of the Model Development Tool subproject. Further information on Eclipse, Papyrus, and GITHUB can be found at:

http://www.eclipse.org/eclipse/

http://www.eclipse.org/papyrus/

http://www.eclispse.org/GITHUB.

An understanding of Eclipse is helpful, but not required; however, familiarity with Papyrus is recommended if you would like to integrate other aspects of a UML model with your NIEM model. Please note that this tutorial only addresses the NIEM UML profile in Papyrus. Papyrus tutorials and user documentation can be found at http://www.eclipse.org/papyrus/usersCorner/usersCornerIndex.php.

Installation

There are options for installing the tool. One is to compile the source, but this is not recommended unless you plan to make changes to the code. The other options involve downloading the proper file for your operating system from github: https://www.github.com and the needed files will install everything for you, including Eclipse. Or you can do the following:

1. Download Java 7 – this is required before moving onto the next step.
2. Download and install Eclipse 4.3 (Kepler) (versions prior to Eclipse 4.3 will not work). Downloading the "Eclipse Standard" is sufficient to support the NIEM Tool. http://www.eclipse.org/downloads
3. Download the NIEM UML Plugin and save it to a memorable place on your computer
4. Install the plugin in Eclipse
5. Open Eclipse
6. In the menu bar click Help -> Install New Software
7. Click the "Add…" button
8. Enter a meaningful name, such as "NIEM UML Plugin"
9. Click the "Archive…" button and select the NIEM UML plugin file that you downloaded during step 3
10. Select the checkbox next to "NIEM UML" and click Next
11. Click "Next" on the install details screen
12. Accept the license agreement and click finish
13. Installation will begin then you'll be prompted to trust a certificate. Click the checkbox next to the certificate name then click OK
14. Restart Eclipse when prompted

The following steps apply, regardless of your approach to installing:

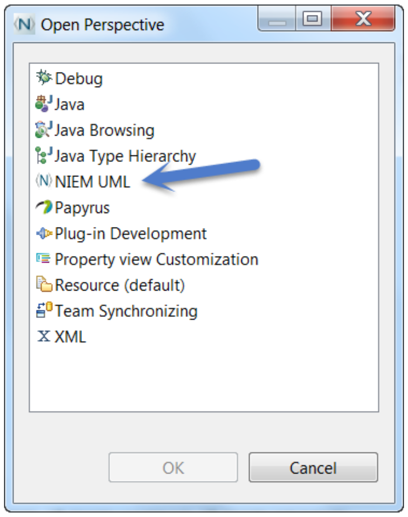
1. Confirm complete installation by being able to change to the NIEM UML Perspective:

a. Click on the Window menu bar option

b. In the project explorer view, right click on a project

c. Select "Open Perspective", then "Other"

d. Choose the "NIEM UML" perspective:



At this point, you have successfully installed the tool and can start creating your exchange model and generate a NIEM Conformant Model Package Description (MPD). An MPD is a defined, collection of organized files that provide the specifications and supporting documentation about the information exchange. The consistent production of an MPD provides benefits of ease of use, consistency and flexibility.

Modeling Process Overview

The high-level steps to create an MPD are:

- Create a general project
- Initialize the NIEM Profile Model
- Model the Extension Packages
- Model the NIEM Subset
- Associate the Extension Classes/Properties with NIEM Subset
- Model the Exchange Package
- Generate the MPD through transformation of the PIM

The scenario for this tutorial is a health exchange between a health care provider, 'Clinician', and a 'Hospital Admitting Unit'. The information to be shared is the services provided by the clinician to the 'Patient' and the medical conditions of a patient that is being referred to the hospital for further health care services.

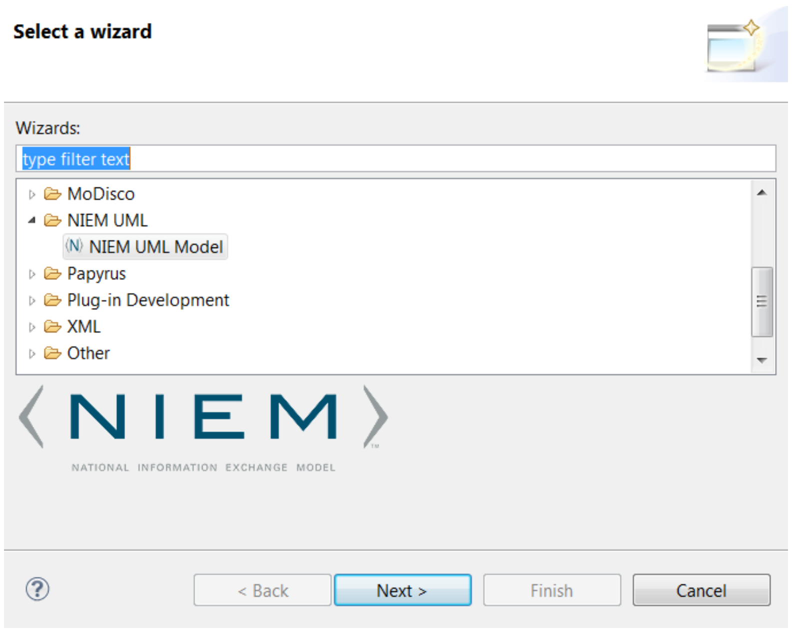
Eclipse Project Creation

First, you'll create a general Eclipse project:

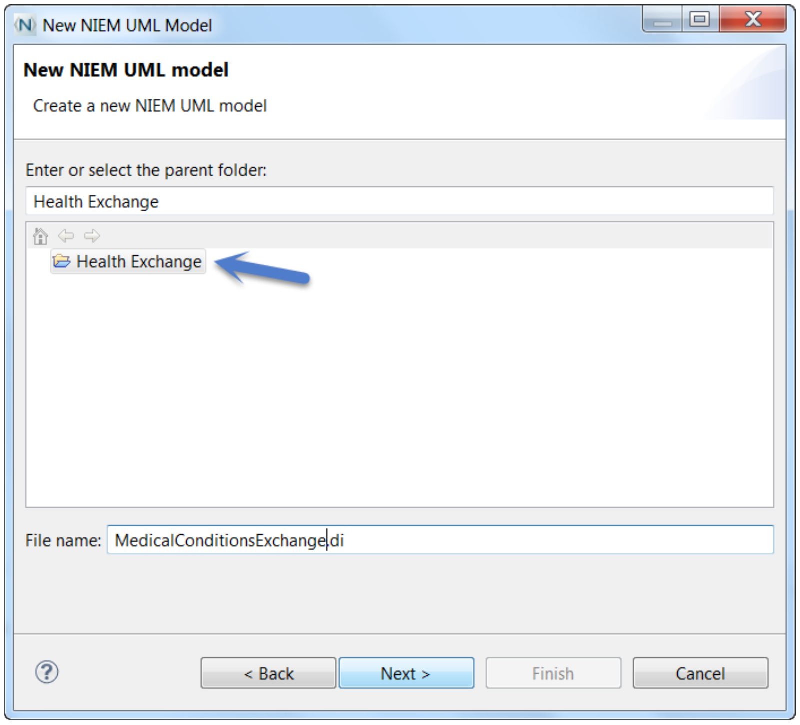
- In the project explorer view, right click in the empty space, click New -> Project
- Select General->Project click Next
- Give the project a name, we'll call our project, 'Health Exchange', and click finish

NIEM Model Initialization

In the project explorer view, right click on the project you just created and select New -> Other and select the NIEM UML Model:



Next, select the project, which we've just created and name your model; we'll call our, 'Medical Conditions Exchange':



New NIEM UML model

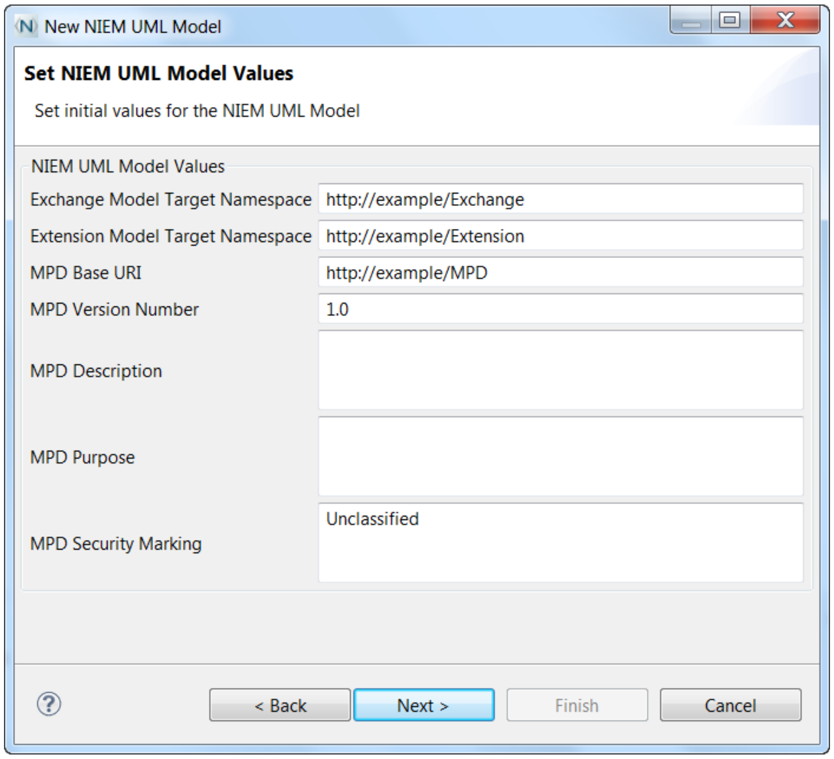
Create a new NIEM UML model

Enter or select the parent folder: Health Exchange

Health Exchange

File name: MedicalConditionsExchange.di

Now, we're going to populate a template that provides metadata values for the model. This information is required to conform to the NIEM Naming and Design Rules (NDRs). Template preferences can be changed under the Window > Preferences > NIEM UML, but for now, we'll just use the default values for now.



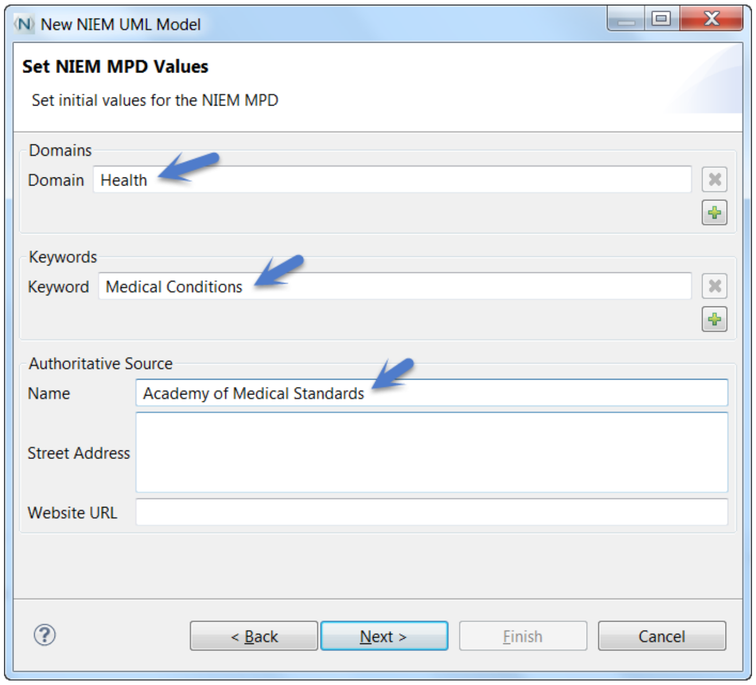
Set NIEM UML Model Values

Set initial values for the NIEM UML Model

NIEM UML Model Values

Exchange Model Target Namespace	http://example/Exchange
Extension Model Target Namespace	http://example/Extension
MPD Base URI	http://example/MPD
MPD Version Number	1.0
MPD Description	
MPD Purpose	
MPD Security Marking	Unclassified

Next we'll set the required values for the MPD: Domain, Keyword, and Name:



Set NIEM MPD Values

Set initial values for the NIEM MPD

Domains

Domain: Health

Keywords

Keyword: Medical Conditions

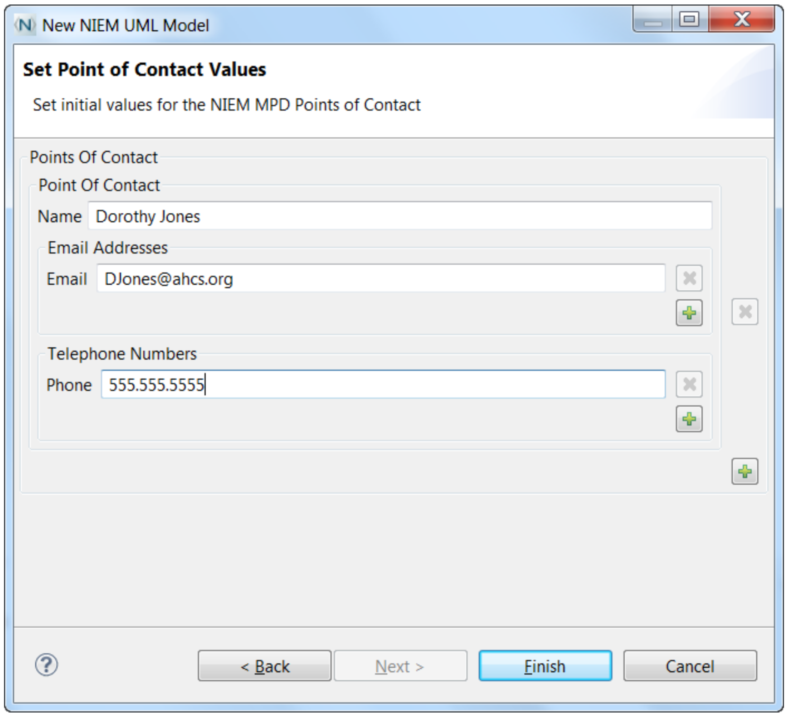
Authoritative Source

Name: Academy of Medical Standards

Street Address:

Website URL:

And the required, Point of Contact Information: Name, Email, and Telephone, then > Finish.



Set Point of Contact Values

Set initial values for the NIEM MPD Points of Contact

Points Of Contact

Point Of Contact

Name: Dorothy Jones

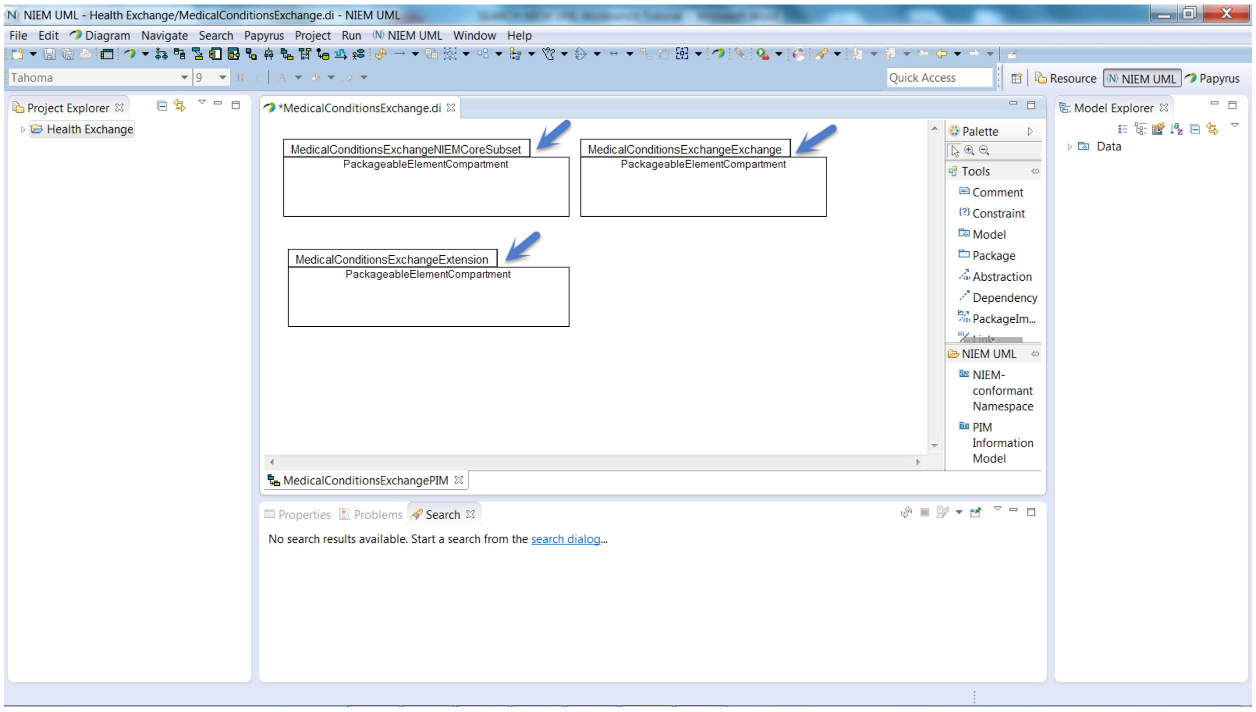
Email Addresses

Email: DJones@ahcs.org

Telephone Numbers

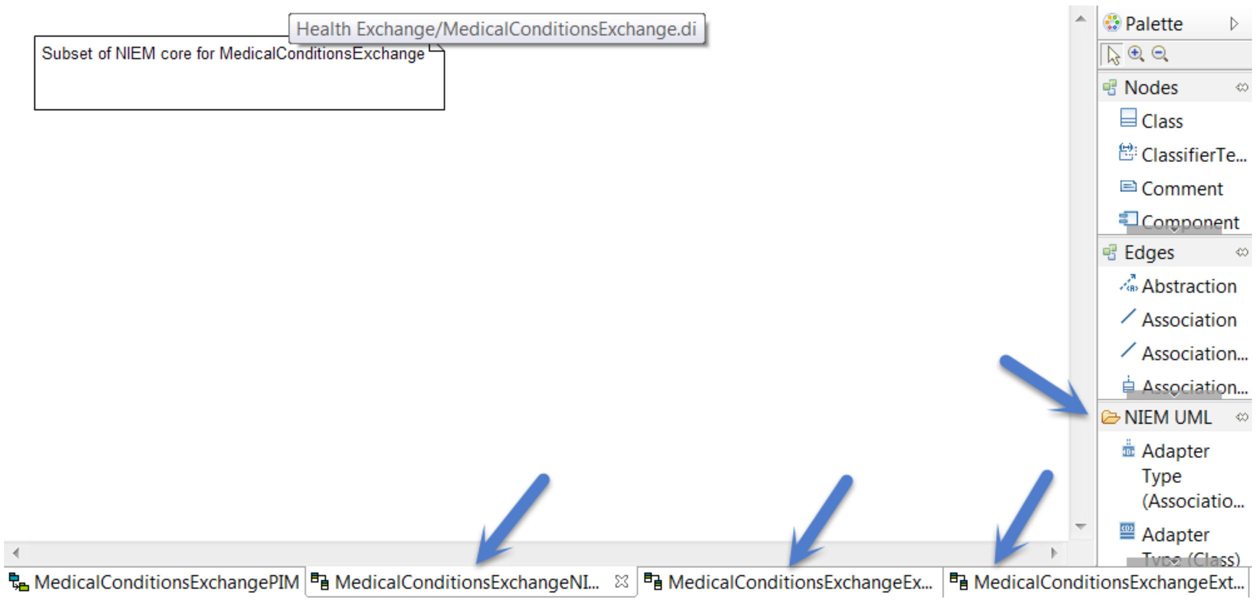
Phone: 555.555.5555

At this point, the tool will generate the initial Platform Independent Model (PIM) package diagram with three packages: a NIEM core subset package, an extension package, and an exchange package, presented in the default view:

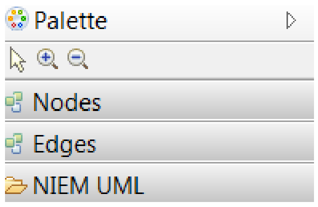

NIEM UML - Health Exchange/MedicalConditionsExchange.di - NIEM UML
File Edit Diagram Navigate Search Papyrus Project Run NIEM UML Window Help
Project Explorer
Health Exchange
*MedicalConditionsExchange.di
MedicalConditionsExchangeNIEMCoreSubset
PackageableElementCompartment
MedicalConditionsExchangeExchange
PackageableElementCompartment
MedicalConditionsExchangeExtension
PackageableElementCompartment
Palette
Tools
Comment
Constraint
Model
Package
Abstraction
Dependency
PackageIm...
NIEM UML
NIEM-conformant Namespace
PIM Information Model
MedicalConditionsExchangePIM
Model Explorer
Data
Resource
NIEM UML
Papyrus
Quick Access
Properties
Problems
Search
No search results available. Start a search from the search dialog...

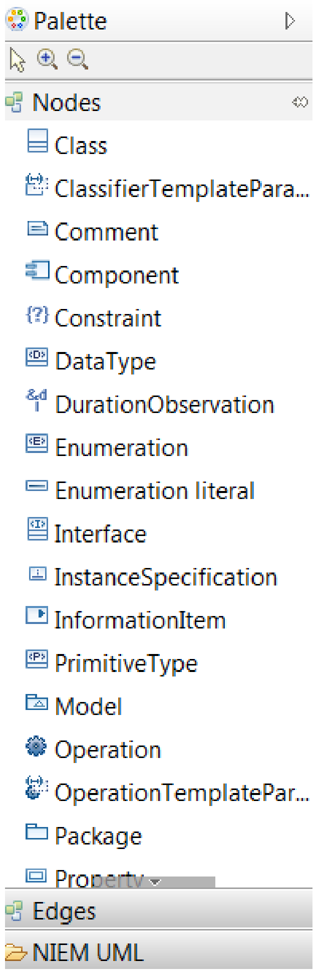
The default view can be changed through the menu bar with: Window > Show View > Other, but for now, we'll stay with the default windows.

Now, we'll generate the Class Diagrams: Double-clicking on each of the PIM packages will produce associated diagrams for the packages, prefaced with the name of our model, MedicalConditionsExchange: The screenshot below is of the NIEM Core subset package; note that the palette includes the 'drawer' of the NIEM UML Profile stereotypes:


Health Exchange/MedicalConditionsExchange.di
Subset of NIEM core for MedicalConditionsExchange
Palette
Nodes
Class
ClassifierTe...
Comment
Component
Edges
Abstraction
Association
Association...
Association...
NIEM UML
Adapter Type (Associatio...
Adapter Type (Class)
MedicalConditionsExchangePIM
MedicalConditionsExchangeNI...
MedicalConditionsExchangeEx...
MedicalConditionsExchangeExt...

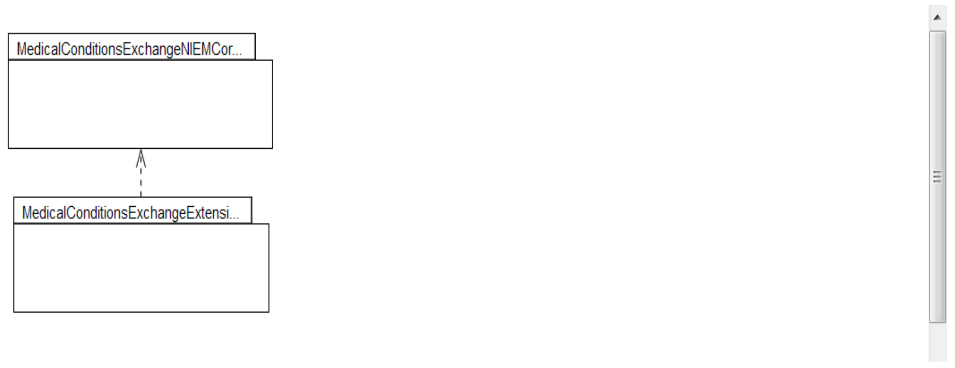
✓ Navigation Tip: In the palette, clicking on the drawer will expand/collapse the drawer:



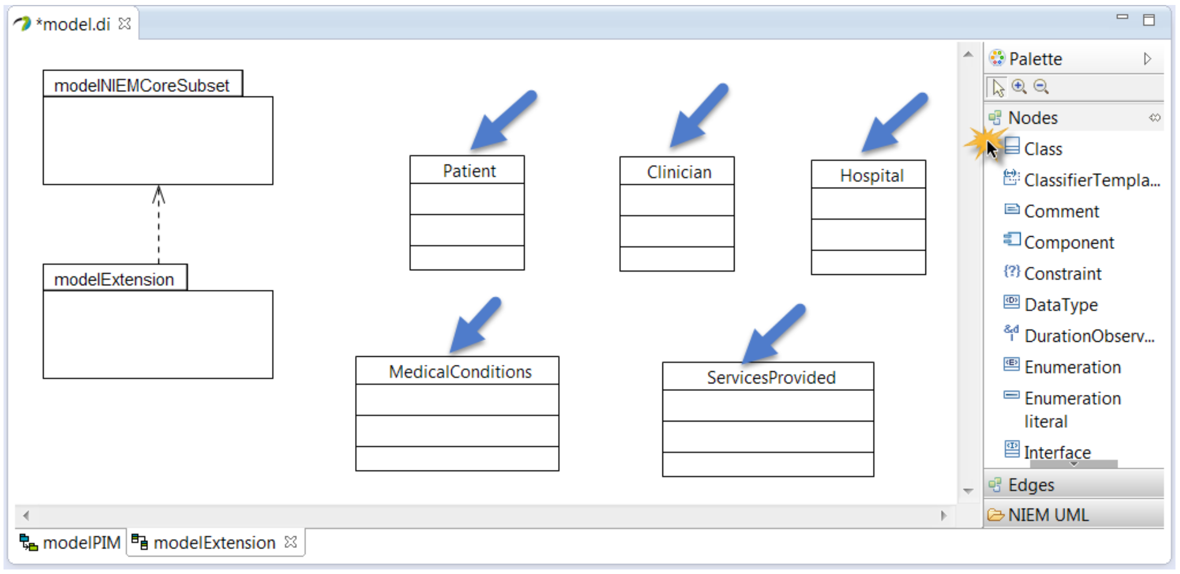


Modeling the Extension Package

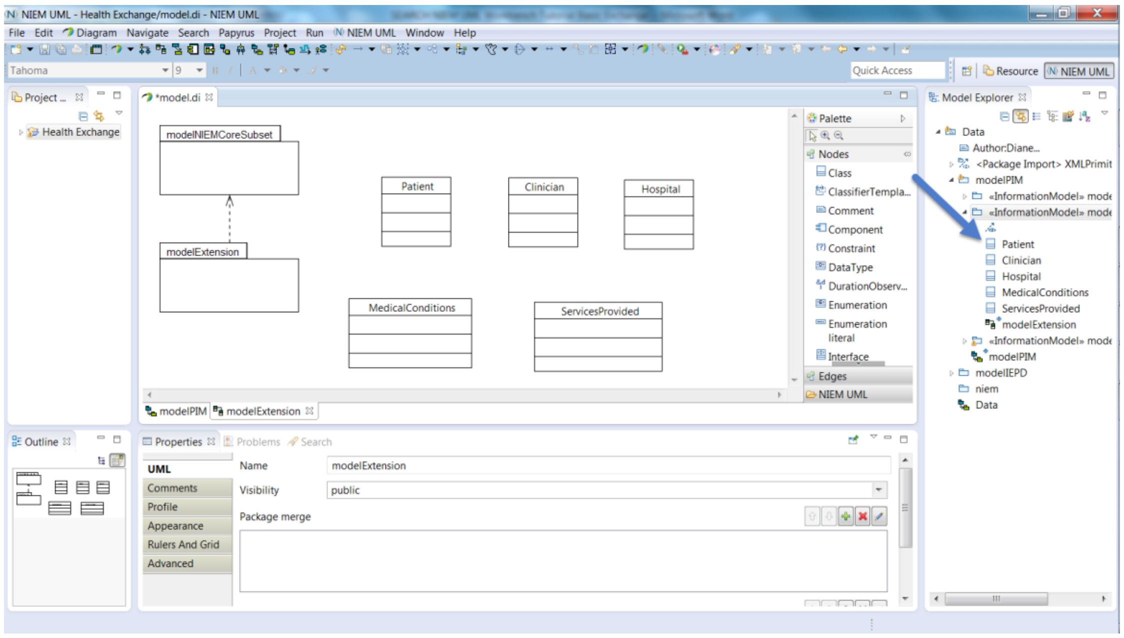
We'll now start modeling our exchange, starting with the extension: the NIEM UML Profile initializes the diagram with two packages:


MedicalConditionsExchangeNIEMCor...
MedicalConditionsExchangeExtensi...

Now we'll add our clinician, patient, and hospital classes. Additionally, we'll add classes for services and medication conditions. These are added as classes from the 'Nodes' drawer.



Also, note that the Model Explorer window reflects our model; the model explorer view contains all the underlying UML elements and can also be used to manipulate the model, but we will continue using the Papyrus main core editor for the moment.



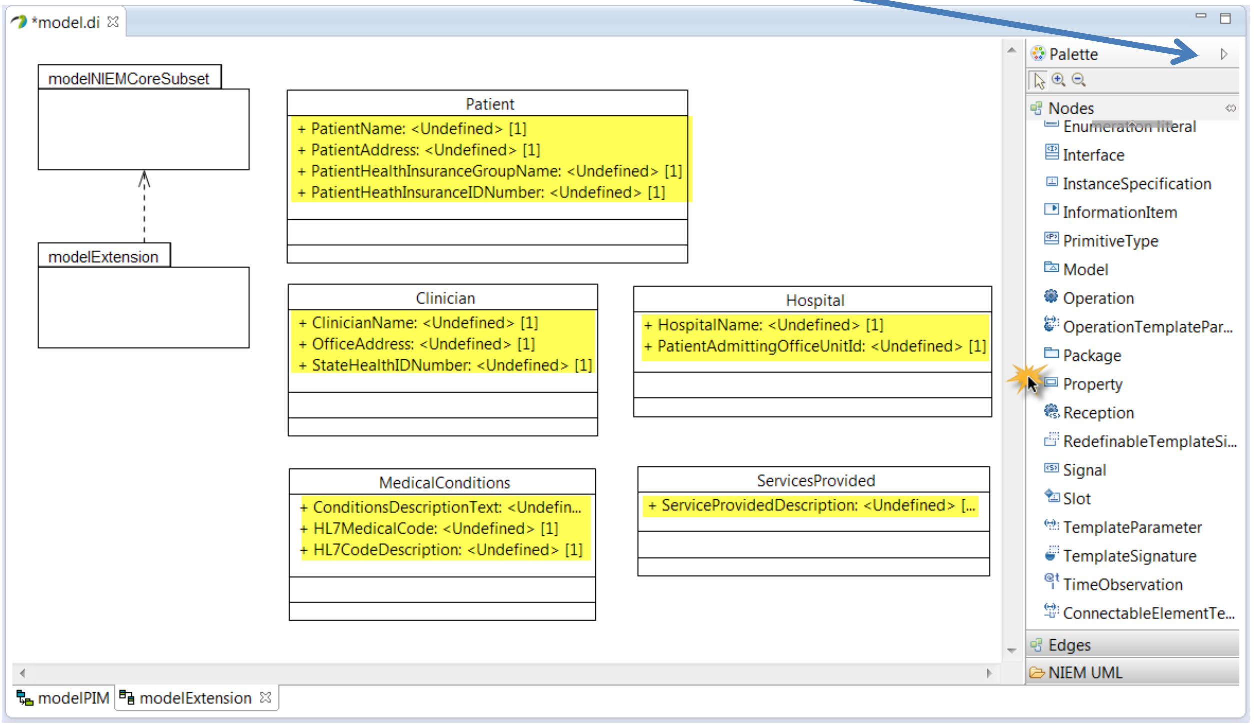
Name: modelExtension

Visibility: public

Package merge

We now add basic properties to our classes, again using the ‘Nodes’ drawer.

✓ Navigation Tip: Use the > to hide/unhide the palette

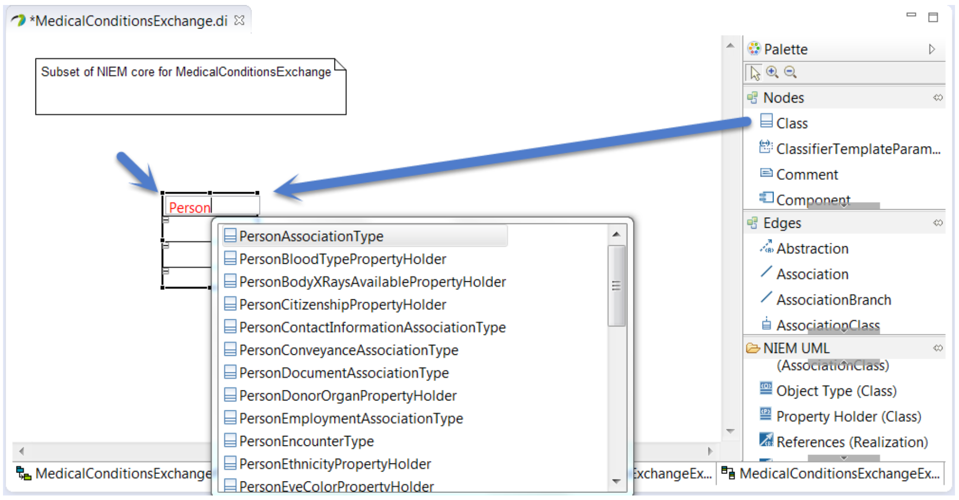

*model.di
modelNIEMCoreSubset
modelExtension
Patient
+ PatientName: <Undefined> [1]
+ PatientAddress: <Undefined> [1]
+ PatientHealthInsuranceGroupName: <Undefined> [1]
+ PatientHeathInsuranceIDNumber: <Undefined> [1]
Clinician
+ ClinicianName: <Undefined> [1]
+ OfficeAddress: <Undefined> [1]
+ StateHealthIDNumber: <Undefined> [1]
Hospital
+ HospitalName: <Undefined> [1]
+ PatientAdmittingOfficeUnitId: <Undefined> [1]
MedicalConditions
+ ConditionsDescriptionText: <Undefin...
+ HL7MedicalCode: <Undefined> [1]
+ HL7CodeDescription: <Undefined> [1]
ServicesProvided
+ ServiceProvidedDescription: <Undefined> [...
Palette
Nodes
Enumeration literal
Interface
InstanceSpecification
InformationItem
PrimitiveType
Model
Operation
OperationTemplatePar...
Package
Property
Reception
RedefinableTemplateSi...
Signal
Slot
TemplateParameter
TemplateSignature
TimeObservation
ConnectableElementTe...
Edges
NIEM UML
modelPIM
modelExtension

Now we will start exploring the NIEM model through the NIEM Core Subset Package.

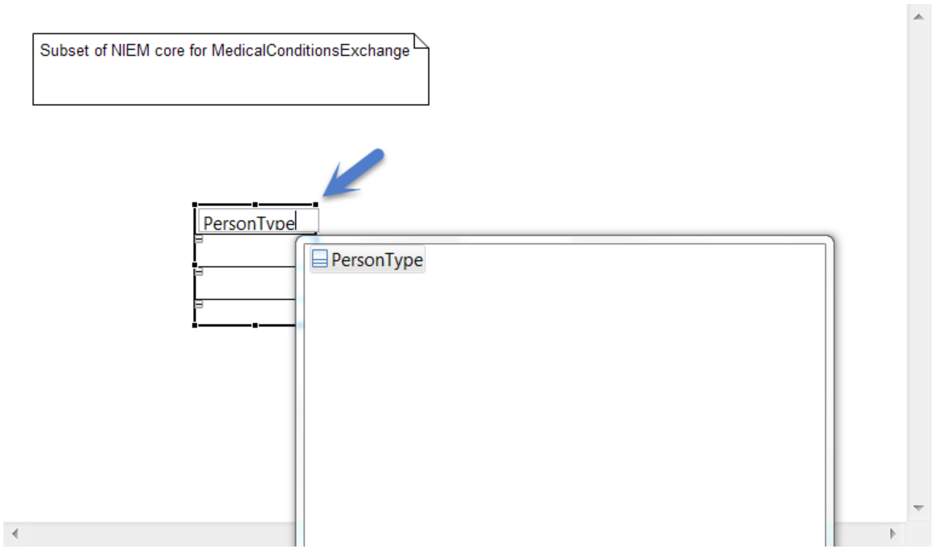
Modeling the NIEM Core Subset Package

Now that we have our extension classes, we’ll see what NIEM information model elements will support our exchange and we open the NIEM Subset package diagram. We know that patients and clinicians are people, so we’ll start with

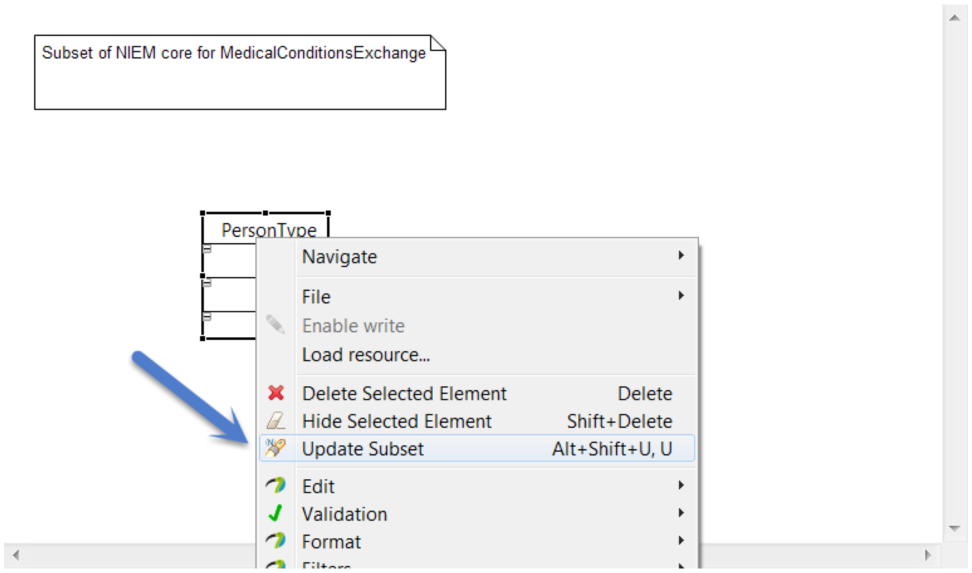
'Person' elements. Since we know that the NIEM model has many elements for person, we'll use the 'look-ahead' method to select our elements. By typing in 'person' in our class name, the tool will display available model elements. The red font indicates that the name is not found in the model, and will change to black when the name is found. The look-ahead pop-up will appear with choices that start with 'Person'.


*MedicalConditionsExchange.di
Subset of NIEM core for MedicalConditionsExchange
Person
PersonAssociationType
PersonBloodTypePropertyHolder
PersonBodyXRaysAvailablePropertyHolder
PersonCitizenshipPropertyHolder
PersonContactInformationAssociationType
PersonConveyanceAssociationType
PersonDocumentAssociationType
PersonDonorOrganPropertyHolder
PersonEmploymentAssociationType
PersonEncounterType
PersonEthnicityPropertyHolder
PersonEyeColorPropertyHolder
Palette
Nodes
Class
ClassifierTemplateParam...
Comment
Component
Edges
Abstraction
Association
AssociationBranch
AssociationClass
NIEM UML
(AssociationClass)
Object Type (Class)
Property Holder (Class)
References (Realization)
MedicalConditionsExchange
MedicalConditionsExchangeEx...

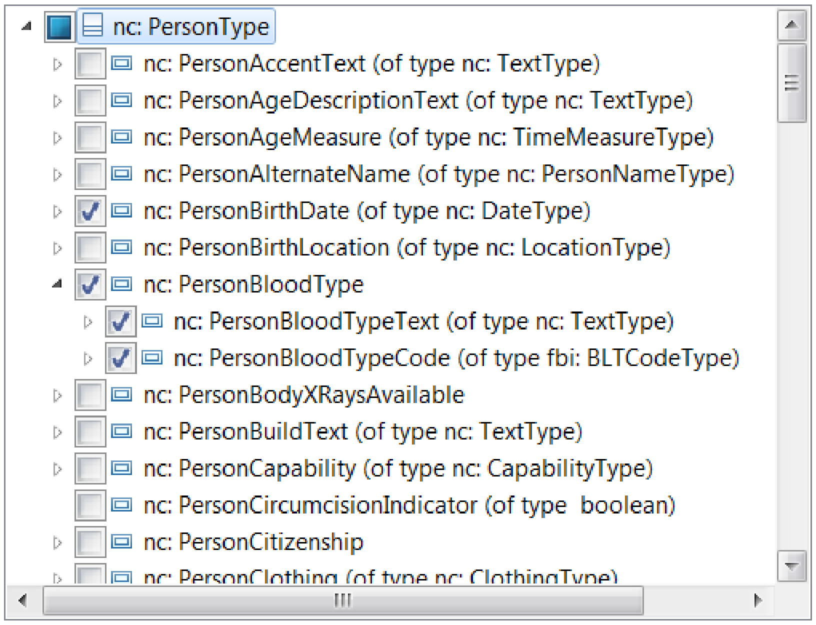
We will use PersonType:



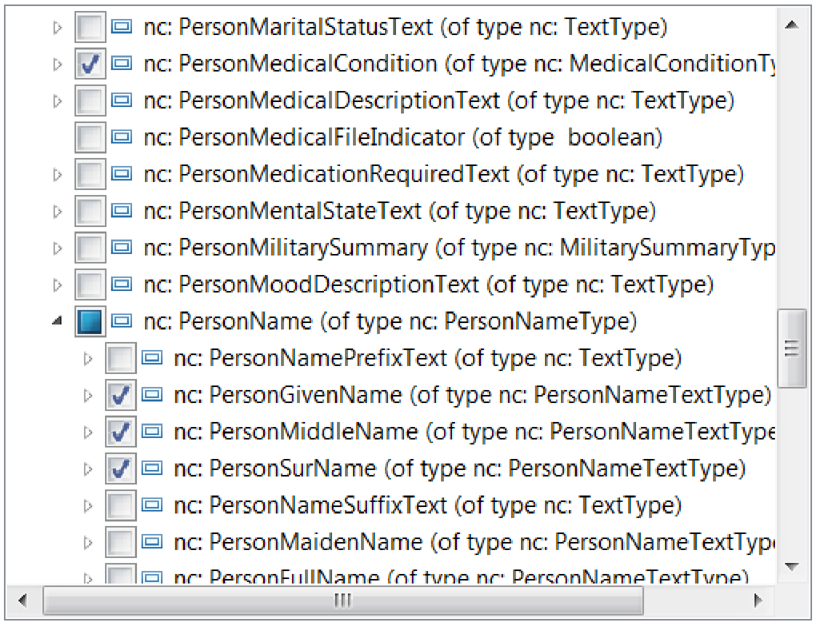
The element is added to the model without any properties, so now we'll need to select what 'PersonType' elements we need to support our exchange. By right-clicking on the 'PersonType' class, we select 'Update Subset' and enter:


Subset of NIEM core for MedicalConditionsExchange
PersonType
Navigate
File
Enable write
Load resource...
Delete Selected Element Delete
Hide Selected Element Shift+Delete
Update Subset Alt+Shift+U, U
Edit
Validation
Format

The window presents navigable, scrollable, and selectable elements for 'PersonType'; we'll select birthdate, blood type, medical conditions.

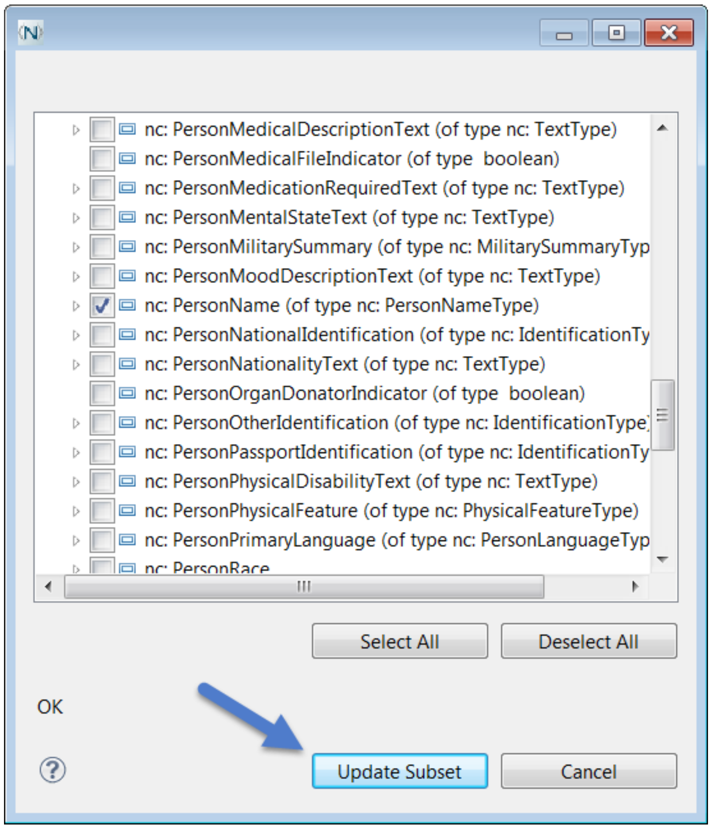


- ■ nc: PersonType
  - ☐ nc: PersonAccentText (of type nc: TextType)
  - ☐ nc: PersonAgeDescriptionText (of type nc: TextType)
  - ☐ nc: PersonAgeMeasure (of type nc: TimeMeasureType)
  - ☐ nc: PersonAlternateName (of type nc: PersonNameType)
  - ✓ nc: PersonBirthDate (of type nc: DateType)
  - ☐ nc: PersonBirthLocation (of type nc: LocationType)
  - ✓ nc: PersonBloodType
    - ✓ nc: PersonBloodTypeText (of type nc: TextType)
    - ✓ nc: PersonBloodTypeCode (of type fbi: BLTCodeType)
  - ☐ nc: PersonBodyXRaysAvailable
  - ☐ nc: PersonBuildText (of type nc: TextType)
  - ☐ nc: PersonCapability (of type nc: CapabilityType)
  - ☐ nc: PersonCircumcisionIndicator (of type boolean)
  - ☐ nc: PersonCitizenship
  - ☐ nc: PersonClothing (of type nc: ClothingType)



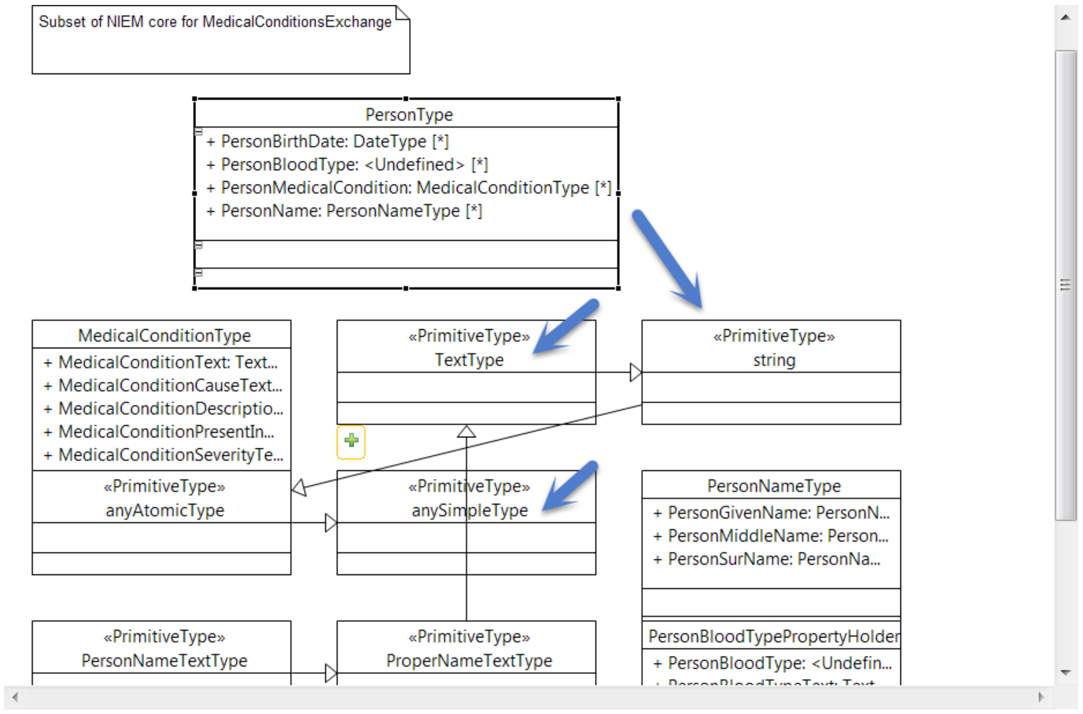
▷ ☐ nc: PersonMaritalStatusText (of type nc: TextType)
▷ ✓ nc: PersonMedicalCondition (of type nc: MedicalConditionTy
▷ ☐ nc: PersonMedicalDescriptionText (of type nc: TextType)
☐ nc: PersonMedicalFileIndicator (of type boolean)
▷ ☐ nc: PersonMedicationRequiredText (of type nc: TextType)
▷ ☐ nc: PersonMentalStateText (of type nc: TextType)
▷ ☐ nc: PersonMilitarySummary (of type nc: MilitarySummaryTyp
▷ ☐ nc: PersonMoodDescriptionText (of type nc: TextType)
◢ ■ nc: PersonName (of type nc: PersonNameType)
  ▷ ☐ nc: PersonNamePrefixText (of type nc: TextType)
  ▷ ✓ nc: PersonGivenName (of type nc: PersonNameTextType)
  ▷ ✓ nc: PersonMiddleName (of type nc: PersonNameTextType
  ▷ ✓ nc: PersonSurName (of type nc: PersonNameTextType)
  ▷ ☐ nc: PersonNameSuffixText (of type nc: TextType)
  ▷ ☐ nc: PersonMaidenName (of type nc: PersonNameTextTyp
  ▷ ☐ nc: PersonFullName (of type nc: PersonNameTextType)

Once we’ve selected our elements, we click on ‘Update Subset’.

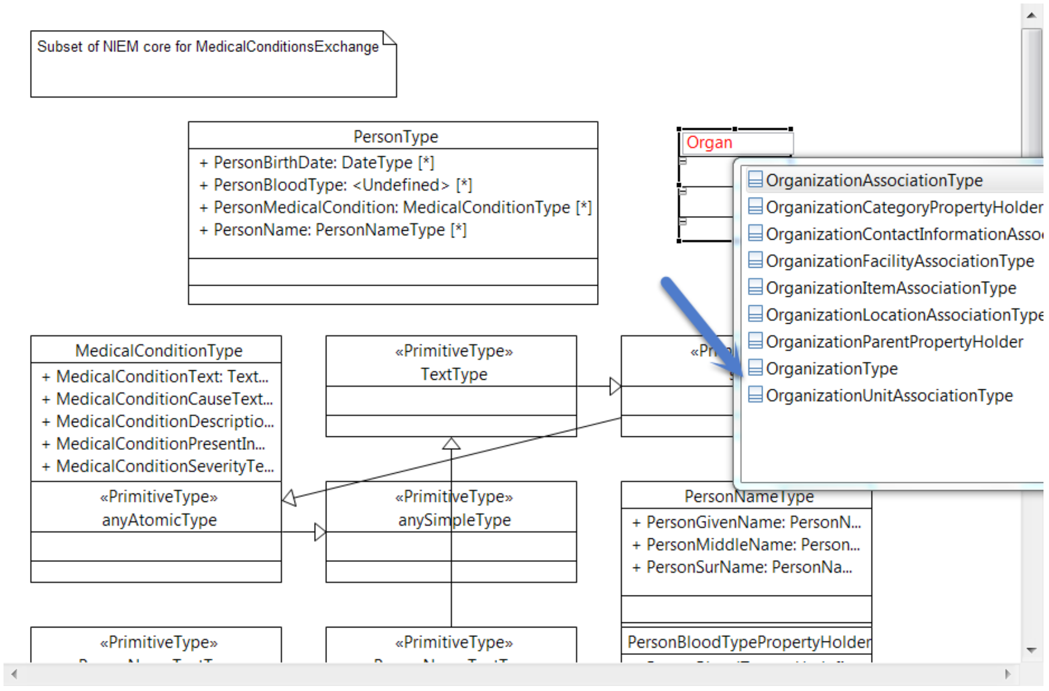


▷ ☐ nc: PersonMedicalDescriptionText (of type nc: TextType)
☐ nc: PersonMedicalFileIndicator (of type boolean)
▷ ☐ nc: PersonMedicationRequiredText (of type nc: TextType)
▷ ☐ nc: PersonMentalStateText (of type nc: TextType)
▷ ☐ nc: PersonMilitarySummary (of type nc: MilitarySummaryTyp
▷ ☐ nc: PersonMoodDescriptionText (of type nc: TextType)
▷ ✓ nc: PersonName (of type nc: PersonNameType)
▷ ☐ nc: PersonNationalIdentification (of type nc: IdentificationTy
▷ ☐ nc: PersonNationalityText (of type nc: TextType)
☐ nc: PersonOrganDonatorIndicator (of type boolean)
▷ ☐ nc: PersonOtherIdentification (of type nc: IdentificationType)
▷ ☐ nc: PersonPassportIdentification (of type nc: IdentificationTy
▷ ☐ nc: PersonPhysicalDisabilityText (of type nc: TextType)
▷ ☐ nc: PersonPhysicalFeature (of type nc: PhysicalFeatureType)
▷ ☐ nc: PersonPrimaryLanguage (of type nc: PersonLanguageTyp
▷ ☐ nc: PersonRace

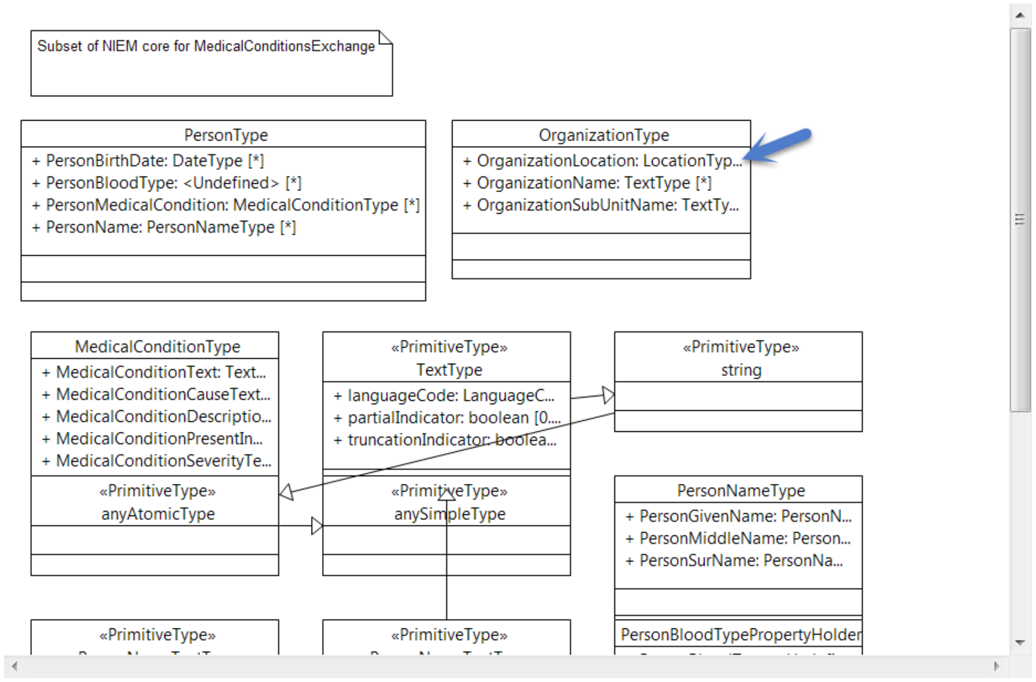
Now, the editor window is displaying not only the elements we’ve selected, but also all the underlying primitive classes needed to support our selected properties and conform to the NIEM NDRs.


Subset of NIEM core for MedicalConditionsExchange
PersonType
+ PersonBirthDate: DateType [*]
+ PersonBloodType: <Undefined> [*]
+ PersonMedicalCondition: MedicalConditionType [*]
+ PersonName: PersonNameType [*]
MedicalConditionType
+ MedicalConditionText: Text...
+ MedicalConditionCauseText...
+ MedicalConditionDescriptio...
+ MedicalConditionPresentIn...
+ MedicalConditionSeverityTe...
«PrimitiveType» TextType
«PrimitiveType» string
«PrimitiveType» anyAtomicType
«PrimitiveType» anySimpleType
PersonNameType
+ PersonGivenName: PersonN...
+ PersonMiddleName: Person...
+ PersonSurName: PersonNa...
«PrimitiveType» PersonNameTextType
«PrimitiveType» ProperNameTextType
PersonBloodTypePropertyHolder
+ PersonBloodType: <Undefin...

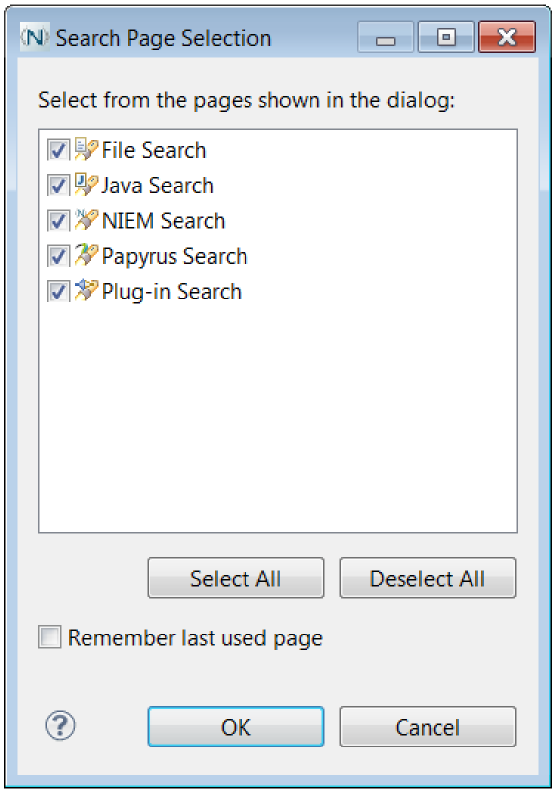
For the moment, this seems to be enough information to support 'Patient' and 'Clinician' information. For the hospital, we know that the NIEM model supports organizations, so we'll do the same technique as we did for 'Person'. As we view the selections, we see that NIEM offers 'OrganizationType', which we will use to represent our hospital.


Subset of NIEM core for MedicalConditionsExchange
PersonType
+ PersonBirthDate: DateType [*]
+ PersonBloodType: <Undefined> [*]
+ PersonMedicalCondition: MedicalConditionType [*]
+ PersonName: PersonNameType [*]
Organ
OrganizationAssociationType
OrganizationCategoryPropertyHolder
OrganizationContactInformationAsso
OrganizationFacilityAssociationType
OrganizationItemAssociationType
OrganizationLocationAssociationType
OrganizationParentPropertyHolder
OrganizationType
OrganizationUnitAssociationType
MedicalConditionType
+ MedicalConditionText: Text...
+ MedicalConditionCauseText...
+ MedicalConditionDescriptio...
+ MedicalConditionPresentIn...
+ MedicalConditionSeverityTe...
«PrimitiveType» TextType
«PrimitiveType» anyAtomicType
«PrimitiveType» anySimpleType
PersonNameType
+ PersonGivenName: PersonN...
+ PersonMiddleName: Person...
+ PersonSurName: PersonNa...
«PrimitiveType»
«PrimitiveType»
PersonBloodTypePropertyHolder

Now, as we did for 'PersonType' we will traverse the subset, select support for the hospital name, location, and a sub unit name for the admitting unit, and update the subset.


Subset of NIEM core for MedicalConditionsExchange
PersonType
+ PersonBirthDate: DateType [*]
+ PersonBloodType: <Undefined> [*]
+ PersonMedicalCondition: MedicalConditionType [*]
+ PersonName: PersonNameType [*]
OrganizationType
+ OrganizationLocation: LocationTyp...
+ OrganizationName: TextType [*]
+ OrganizationSubUnitName: TextTy...
MedicalConditionType
+ MedicalConditionText: Text...
+ MedicalConditionCauseText...
+ MedicalConditionDescriptio...
+ MedicalConditionPresentIn...
+ MedicalConditionSeverityTe...
«PrimitiveType» TextType
+ languageCode: LanguageC...
+ partialIndicator: boolean [0....
+ truncationIndicator: boolea...
«PrimitiveType» string
«PrimitiveType» anyAtomicType
«PrimitiveType» anySimpleType
PersonNameType
+ PersonGivenName: PersonN...
+ PersonMiddleName: Person...
+ PersonSurName: PersonNa...
«PrimitiveType»
«PrimitiveType»
PersonBloodTypePropertyHolder

Now we need to consider what is needed to support ‘ServicesProvided’. Since we aren’t as familiar as we are with people and organizations, we will use the ‘Search’ capability. We know that providing a service is an activity, so we’ll start there. The NIEM model ‘Search’ capability is through the ‘Search’ menu bar selection and provides options for results. When you first initialize a tool, the default is all pages and you will receive a window to ‘Customize’ your search.

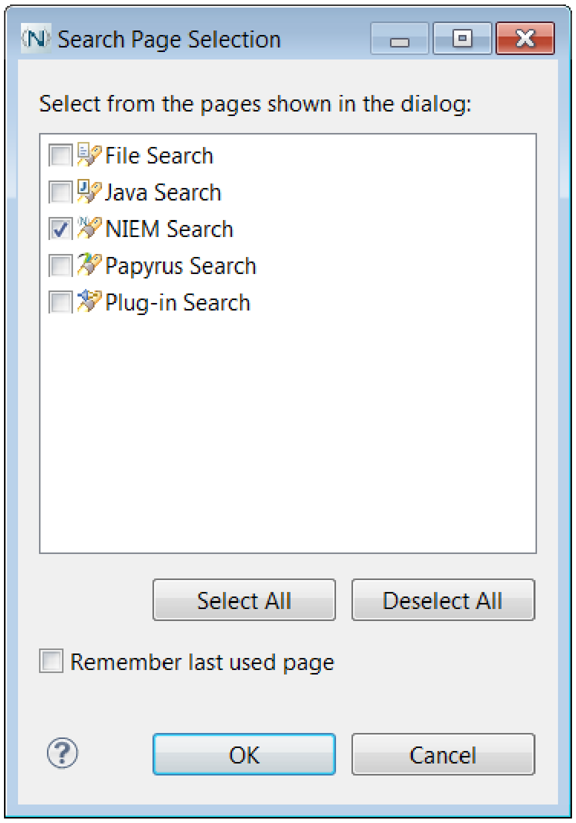

Search Page Selection

Select from the pages shown in the dialog:

- [✓] File Search
- [✓] Java Search
- [✓] NIEM Search
- [✓] Papyrus Search
- [✓] Plug-in Search

[ ] Remember last used page

For our purposes, we will deselect ‘File’, ‘Java’, ‘Papyrus’, and ‘Plug-in’.



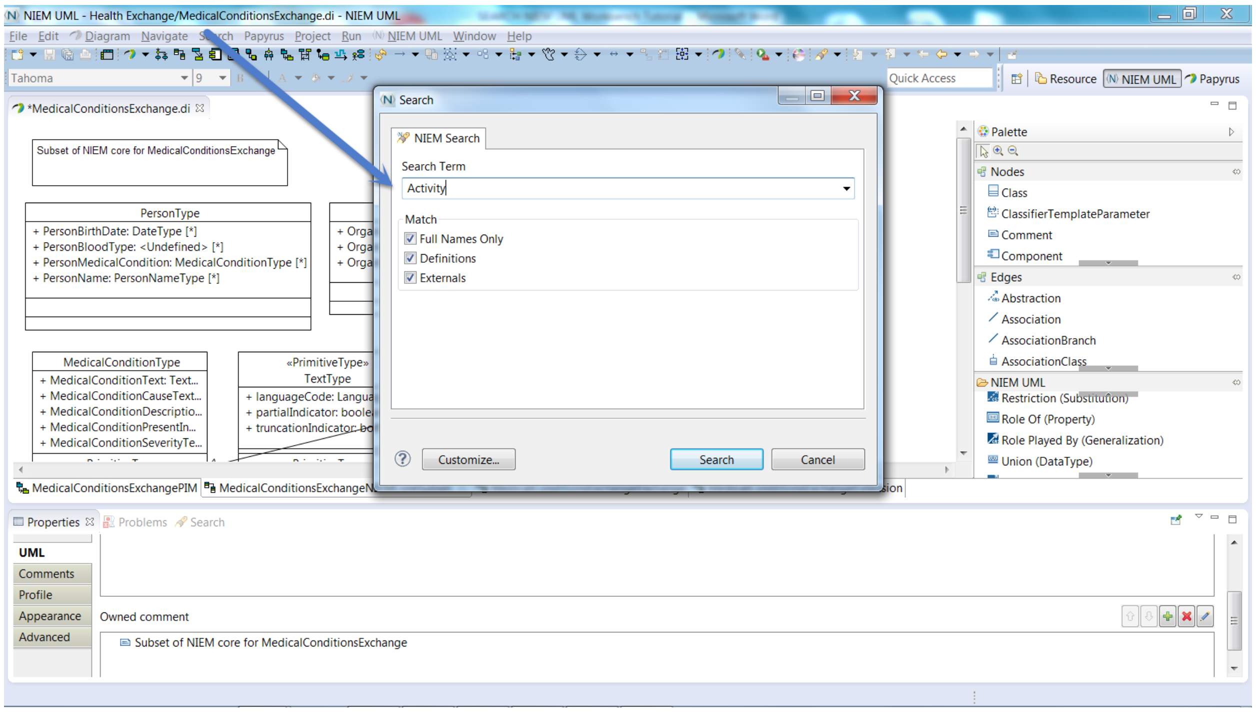
Search Page Selection

Select from the pages shown in the dialog:

- [ ] File Search
- [ ] Java Search
- [✓] NIEM Search
- [ ] Papyrus Search
- [ ] Plug-in Search

[ ] Remember last used page

We search on ‘Activity’,



Search

NIEM Search

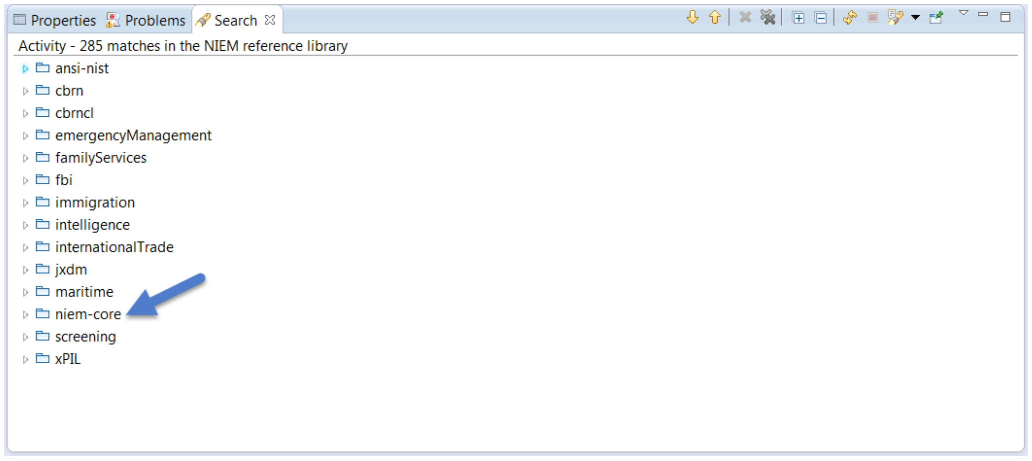
Search Term: Activity

Match

- [✓] Full Names Only
- [✓] Definitions
- [✓] Externals

NIEM UML - Health Exchange/MedicalConditionsExchange.di - NIEM UML
File Edit Diagram Navigate Search Papyrus Project Run NIEM UML Window Help
*MedicalConditionsExchange.di
Subset of NIEM core for MedicalConditionsExchange
PersonType
MedicalConditionType
«PrimitiveType» TextType
Customize...
Search
Cancel
Palette
Nodes
Edges
NIEM UML
Properties
Problems
UML
Comments
Profile
Appearance
Advanced
Owned comment

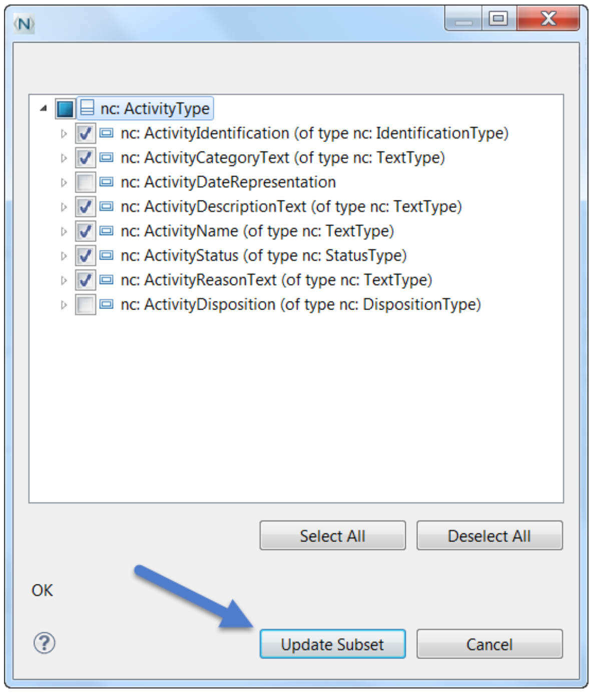
and receive 285 matches from several domains. However, it is a good practice to look in NIEM core first.



Activity - 285 matches in the NIEM reference library

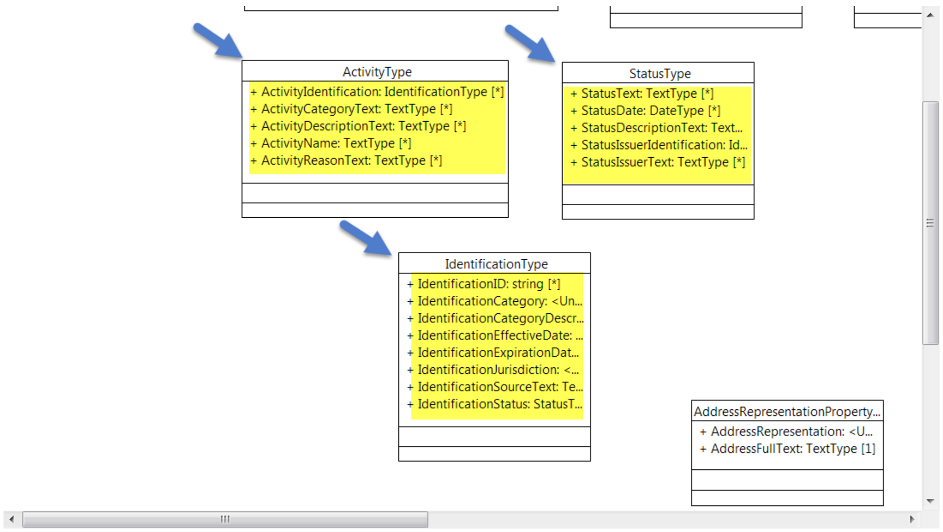
- ansi-nist
- cbrn
- cbrncl
- emergencyManagement
- familyServices
- fbi
- immigration
- intelligence
- internationalTrade
- jxdm
- maritime
- niem-core
- screening
- xPIL

We will expand 'ActivityType', select our elements, and 'Update Subset'.



- nc: ActivityType
  - [✓] nc: ActivityIdentification (of type nc: IdentificationType)
  - [✓] nc: ActivityCategoryText (of type nc: TextType)
  - [ ] nc: ActivityDateRepresentation
  - [✓] nc: ActivityDescriptionText (of type nc: TextType)
  - [✓] nc: ActivityName (of type nc: TextType)
  - [✓] nc: ActivityStatus (of type nc: StatusType)
  - [✓] nc: ActivityReasonText (of type nc: TextType)
  - [ ] nc: ActivityDisposition (of type nc: DispositionType)

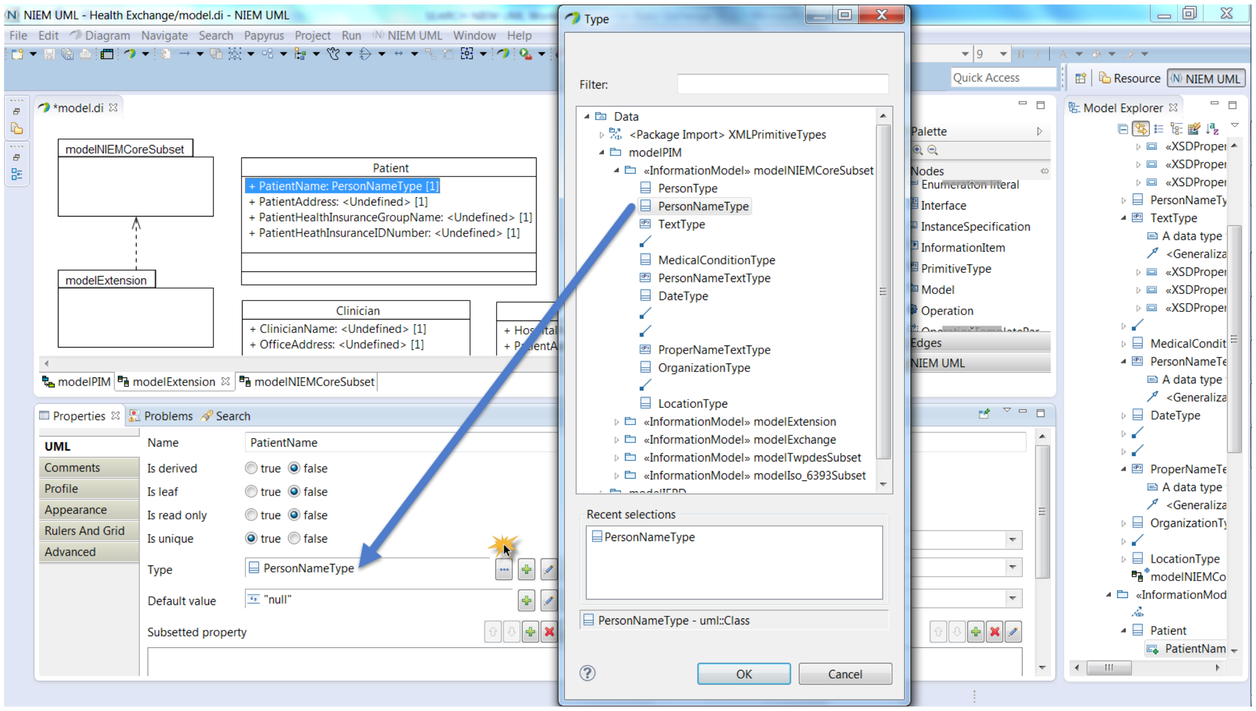
Note that when we updated 'ActivityType' – from the properties we selected, 'IdentificationType' and 'StatusType' were generated as empty classes, so we also update those classes.


ActivityType
+ ActivityIdentification: IdentificationType [*]
+ ActivityCategoryText: TextType [*]
+ ActivityDescriptionText: TextType [*]
+ ActivityName: TextType [*]
+ ActivityReasonText: TextType [*]
StatusType
+ StatusText: TextType [*]
+ StatusDate: DateType [*]
+ StatusDescriptionText: Text...
+ StatusIssuerIdentification: Id...
+ StatusIssuerText: TextType [*]
IdentificationType
+ IdentificationID: string [*]
+ IdentificationCategory: <Un...
+ IdentificationCategoryDescr...
+ IdentificationEffectiveDate: ...
+ IdentificationExpirationDat...
+ IdentificationJurisdiction: <...
+ IdentificationSourceText: Te...
+ IdentificationStatus: StatusT...
AddressRepresentationProperty...
+ AddressRepresentation: <U...
+ AddressFullText: TextType [1]

Associating the Extension Schema Properties to the NIEM Core Subset Package

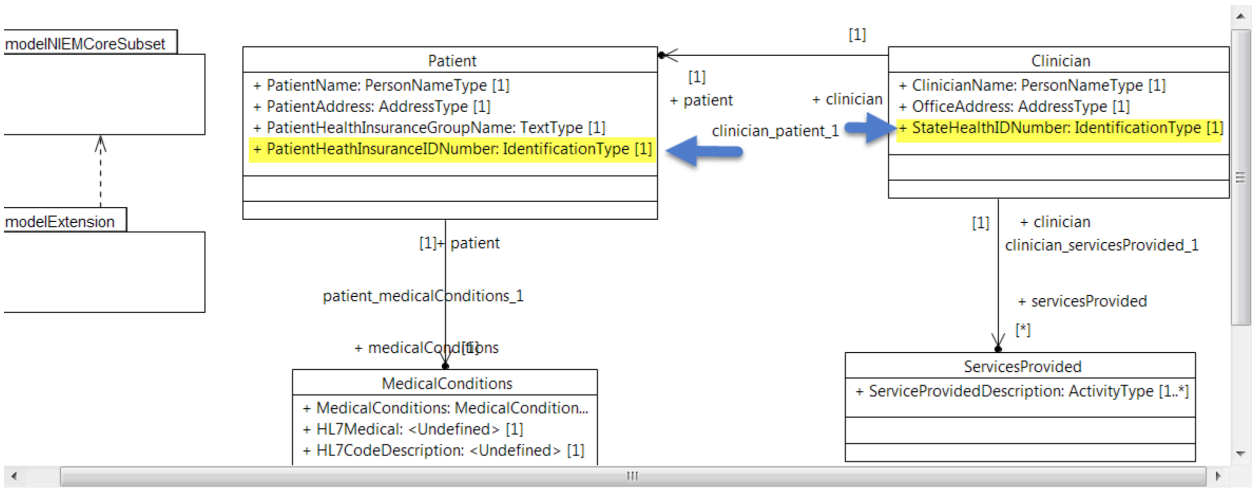
At this point, we feel that the NIEM information model will support our requirements, so we need to associate our extension schema classes to our NIEM core subset schema classes and return to our extension schema diagram.

We select the 'PersonName' property in Patient' and in the 'Properties' window, click on '…' in the 'Type' field. This will allow us to navigate through the NIEM subset classes that we've just defined. A window is presented with a navigable and scrollable, window, in addition to being able to filter through the options. This is invaluable for larger models. We select 'PersonNameType' to associate with our 'PatientName'.

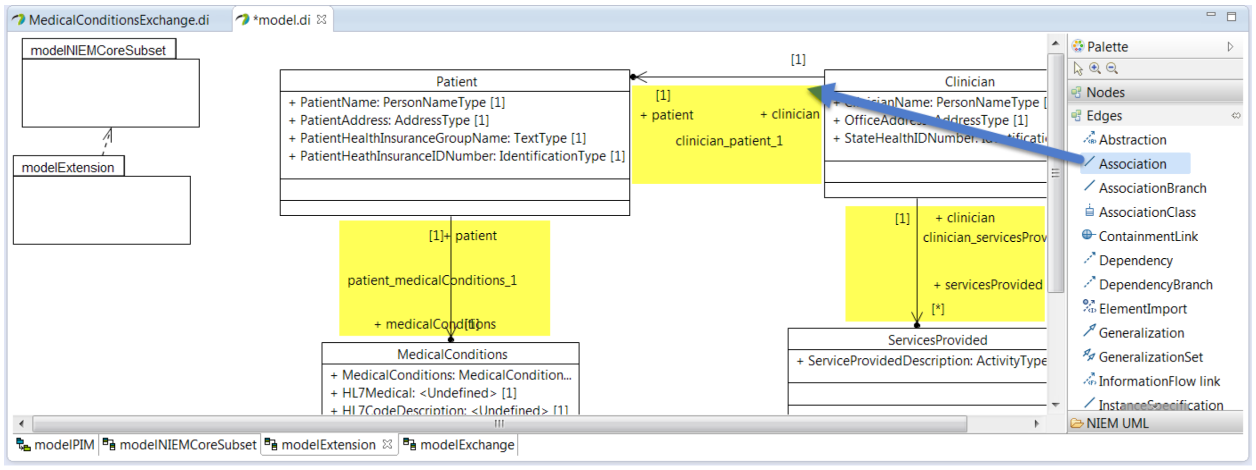


Name: PatientName
Is derived: true [✓ false]
Is leaf: true [✓ false]
Is read only: true [✓ false]
Is unique: [✓ true] false
Type: PersonNameType
Default value: "null"
Subsetted property

Similarly, we associate 'PatientAddress' with 'AddressType', 'PatientHealthInsuranceGroupName' as a 'TextType', and 'PatientHealthInsuranceIDNumber' with 'IdentificationType'.

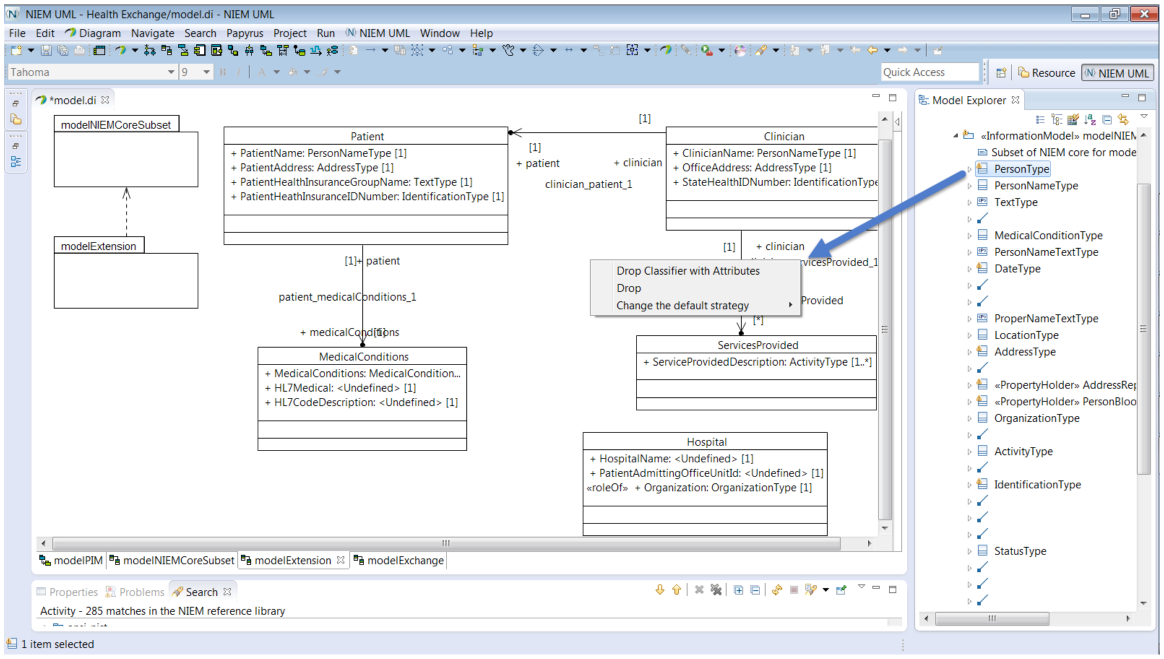

modelNIEMCoreSubset
modelExtension
Patient
+ PatientName: PersonNameType [1]
+ PatientAddress: AddressType [1]
+ PatientHealthInsuranceGroupName: TextType [1]
+ PatientHeathInsuranceIDNumber: IdentificationType [1]
[1]
+ patient
+ clinician
clinician_patient_1
[1]
Clinician
+ ClinicianName: PersonNameType [1]
+ OfficeAddress: AddressType [1]
+ StateHealthIDNumber: IdentificationType [1]
[1]+ patient
patient_medicalConditions_1
+ medicalConditions [1]
MedicalConditions
+ MedicalConditions: MedicalCondition...
+ HL7Medical: <Undefined> [1]
+ HL7CodeDescription: <Undefined> [1]
[1] + clinician
clinician_servicesProvided_1
+ servicesProvided
[*]
ServicesProvided
+ ServiceProvidedDescription: ActivityType [1..*]

And we add associations using 'Association' edge.

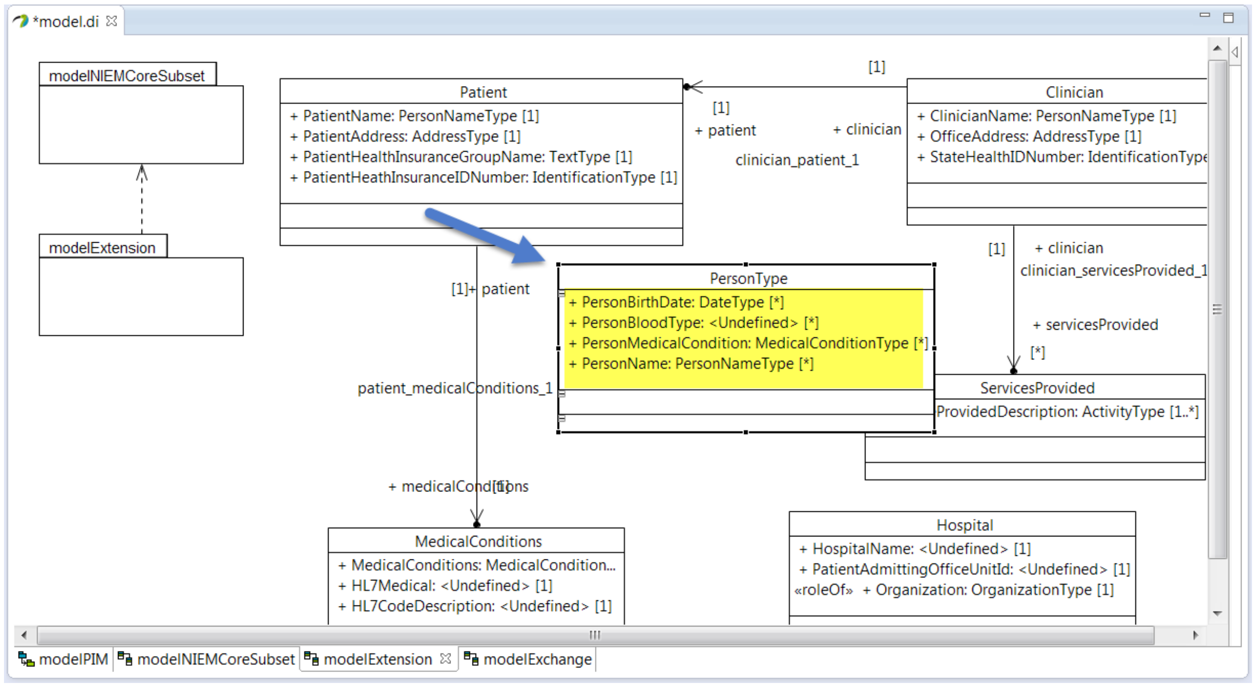


We now need to associate our NIEM classes with our extension classes. First we'll associate the NIEM subset 'PersonType' with 'Patient'. We select 'PersonType' in the model explorer, hold & drag into core editor window, and the options pop-up is presented to 'Drop with Attributes', 'Drop', or 'Change Strategy'.

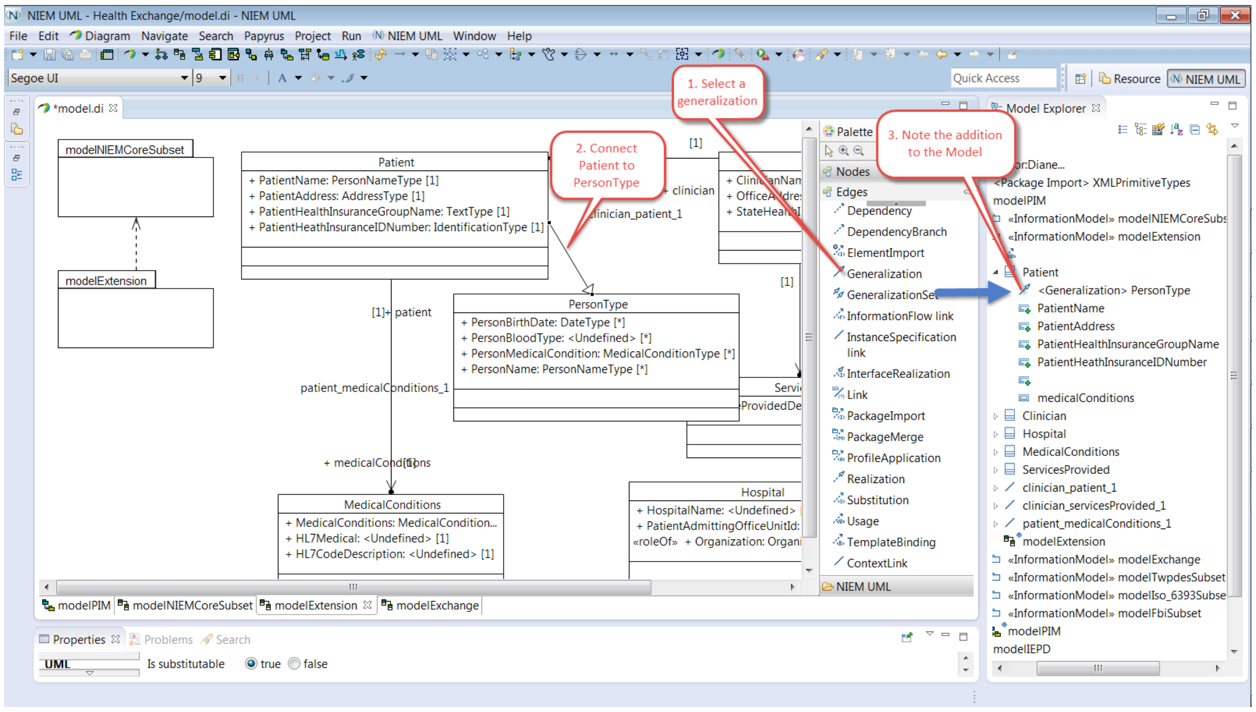
- ✓ Navigation Tip: 'Change Strategy' allows you to set a default to always 'Drop with Attributes' or 'Drop'.



We will 'Drop with Attributes'.


PersonType
+ PersonBirthDate: DateType [*]
+ PersonBloodType: <Undefined> [*]
+ PersonMedicalCondition: MedicalConditionType [*]
+ PersonName: PersonNameType [*]

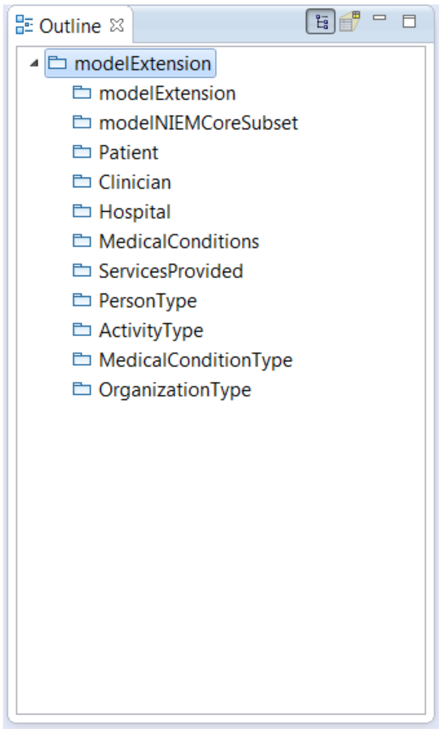
We will now make the generalization relationship between 'PersonType' and 'Patient', to extend the NIEM PersonType with the properties of our 'Patient' class.


1. Select a generalization
2. Connect Patient to PersonType
3. Note the addition to the Model

✓ Navigation Tip: Use the ‘Arrange All’ or ‘Arrange Selection’ to make the class diagram easier to work with, plus it is fun to watch the classes shift around.

We will now make the generalization relationship between ‘PersonType’ and ‘Clinician’, ‘ActivityType’ and ‘ServicesProvided’, and ‘Patient’, to extend the NIEM PersonType with the properties of our ‘Patient’ class.

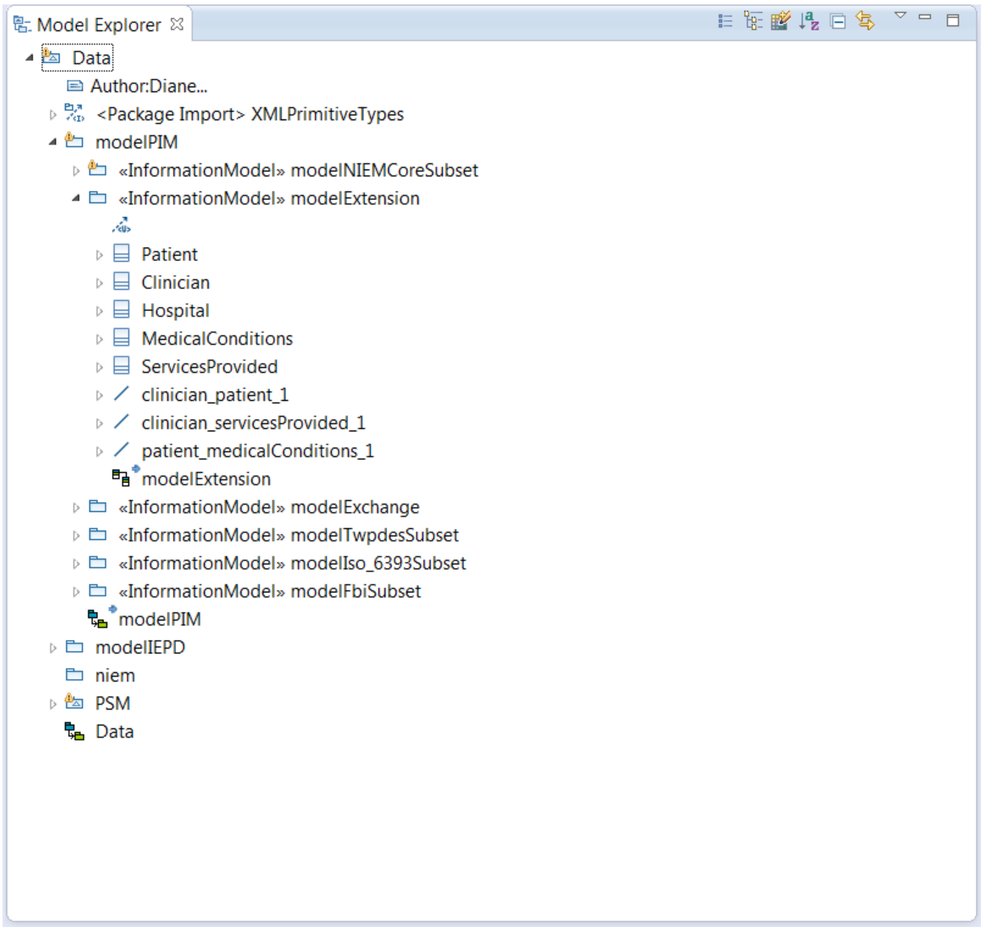
The Outline View for the Extension Package now looks like this:



Outline

- modelExtension
  - modelExtension
  - modelNIEMCoreSubset
  - Patient
  - Clinician
  - Hospital
  - MedicalConditions
  - ServicesProvided
  - PersonType
  - ActivityType
  - MedicalConditionType
  - OrganizationType

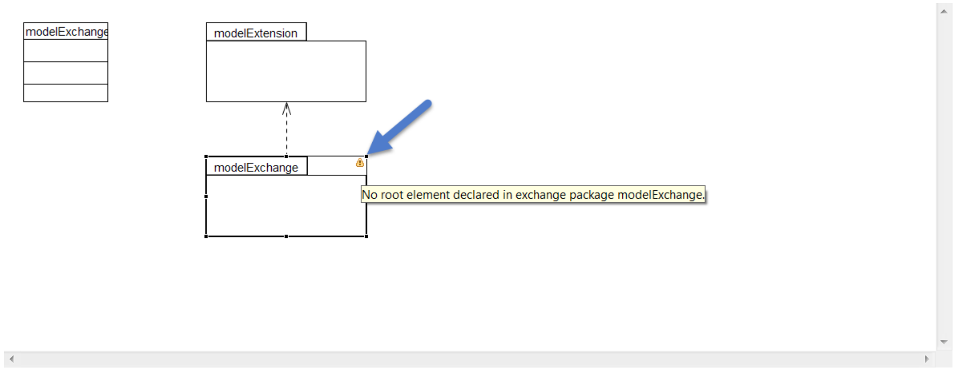
And the Model Explorer view of our Extension Package now looks like this:


Model Explorer

- Data
  - Author:Diane...
  - <Package Import> XMLPrimitiveTypes
  - modelPIM
    - «InformationModel» modelNIEMCoreSubset
    - «InformationModel» modelExtension
      - Patient
      - Clinician
      - Hospital
      - MedicalConditions
      - ServicesProvided
      - clinician_patient_1
      - clinician_servicesProvided_1
      - patient_medicalConditions_1
      - modelExtension
    - «InformationModel» modelExchange
    - «InformationModel» modelTwpdesSubset
    - «InformationModel» modelIso_6393Subset
    - «InformationModel» modelFbiSubset
    - modelPIM
  - modelIEPD
  - niem
  - PSM
  - Data

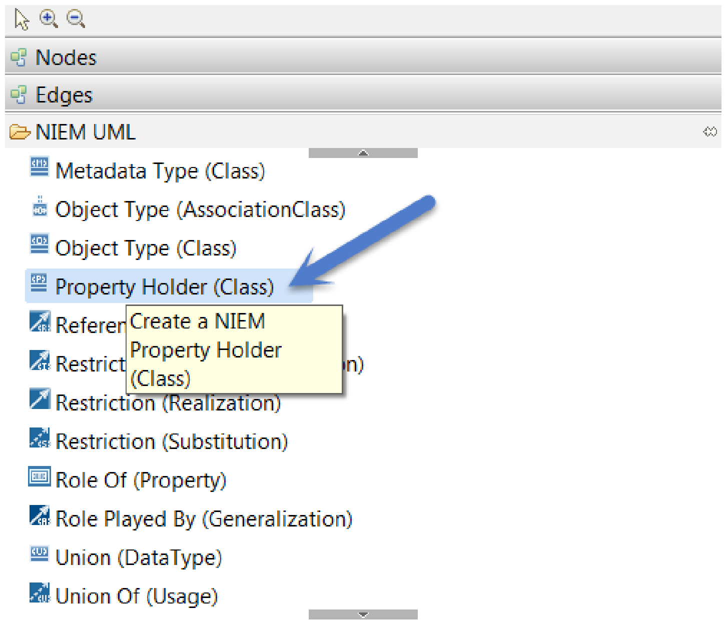
Modeling the NIEM Exchange Package

Now we are going to model the exchange and generate the MPD. For this, we enter the Exchange Diagram which is initialized as below. Note the yellow error icon, indicating that a root element is needed.

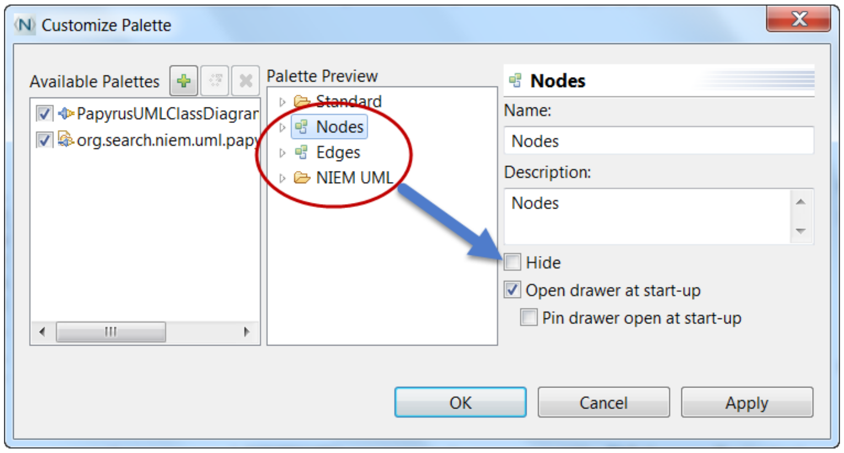


No root element declared in exchange package modelExchange.

For this, we need to use a NIEM stereotype, a ‘Property Holder’ class which will contain the properties to be exchanged. We will call our container ‘MedicalConditionsDocument’.


Nodes
Edges
NIEM UML
Metadata Type (Class)
Object Type (AssociationClass)
Object Type (Class)
Property Holder (Class)
Referen
Create a NIEM Property Holder (Class)
Restrict
n)
Restriction (Realization)
Restriction (Substitution)
Role Of (Property)
Role Played By (Generalization)
Union (DataType)
Union Of (Usage)

✓ Navigation Tip: The palette drawers can be customized to hide elements; right click in the palette, select 'customize' and you will receive the 'Customize Palette' window. Here you can expand any drawer & hide elements that are infrequently used, along with open/close/pin drawer behavior.



Nodes

Name: Nodes

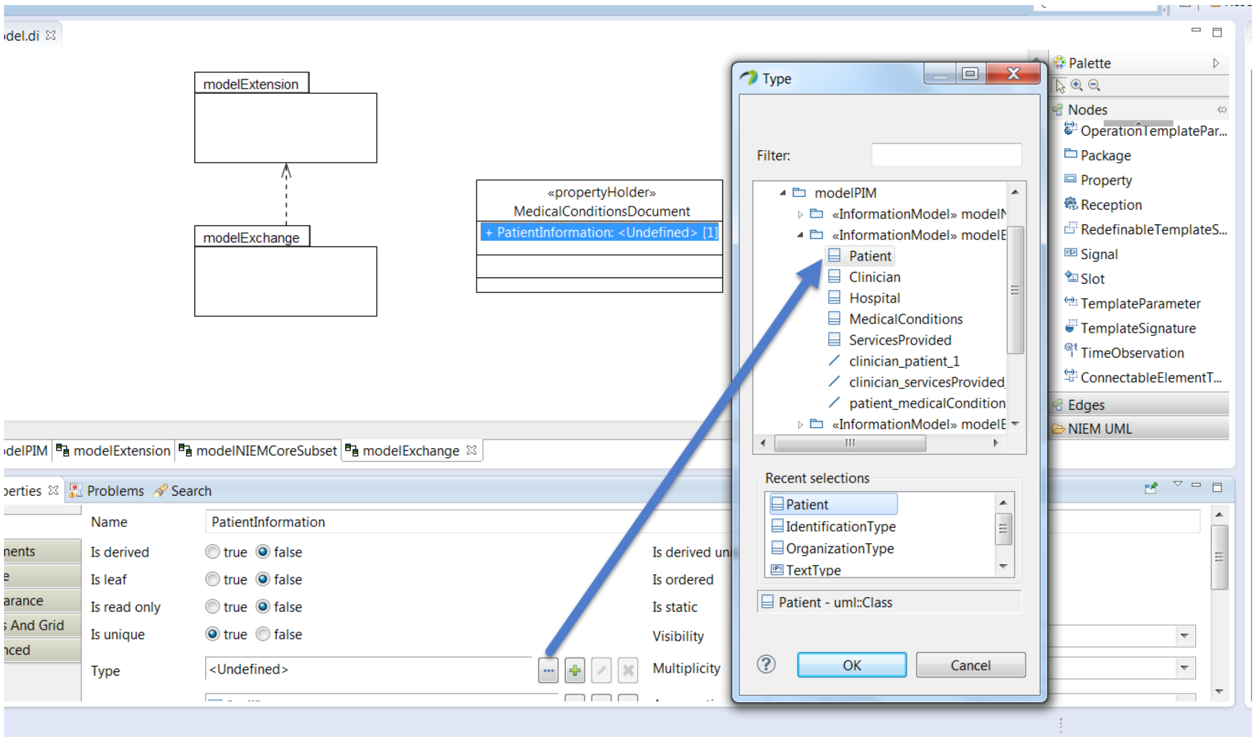
Description: Nodes

☐ Hide

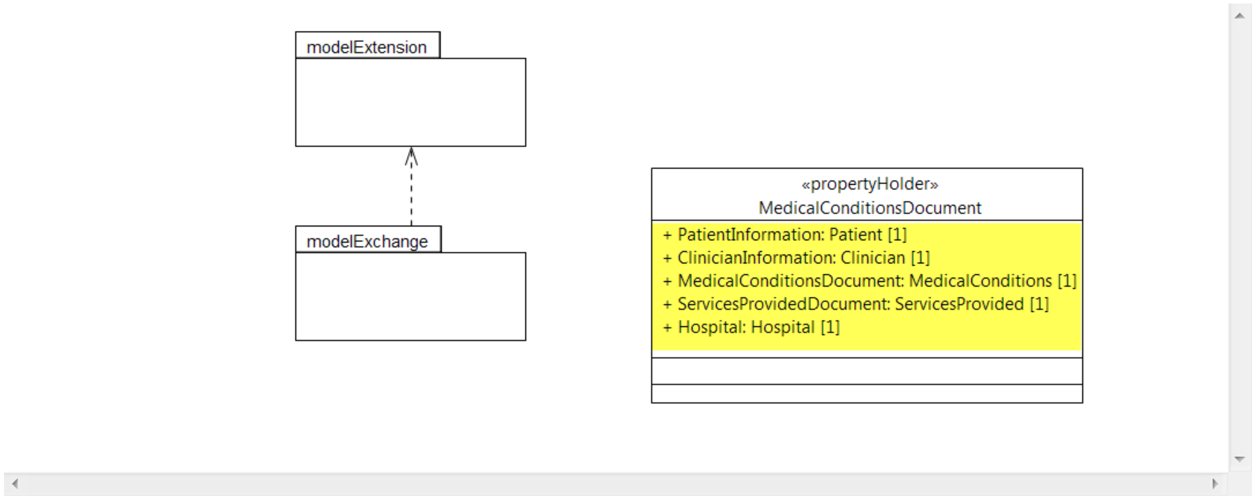
☑ Open drawer at start-up

☐ Pin drawer open at start-up

Now, we will add a property called 'PatientInformation' and define it to our 'Patient Class' in our extension schema.



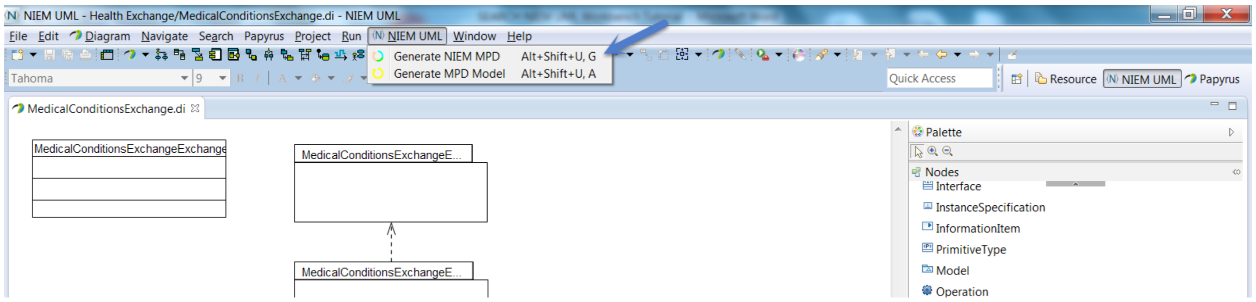
Similarly, we will add ‘ClinicianInformation’, ‘MedicalConditionsDocument’, and ‘ServicesProvidedDocument’.


modelExtension
modelExchange
«propertyHolder»
MedicalConditionsDocument
+ PatientInformation: Patient [1]
+ ClinicianInformation: Clinician [1]
+ MedicalConditionsDocument: MedicalConditions [1]
+ ServicesProvidedDocument: ServicesProvided [1]
+ Hospital: Hospital [1]

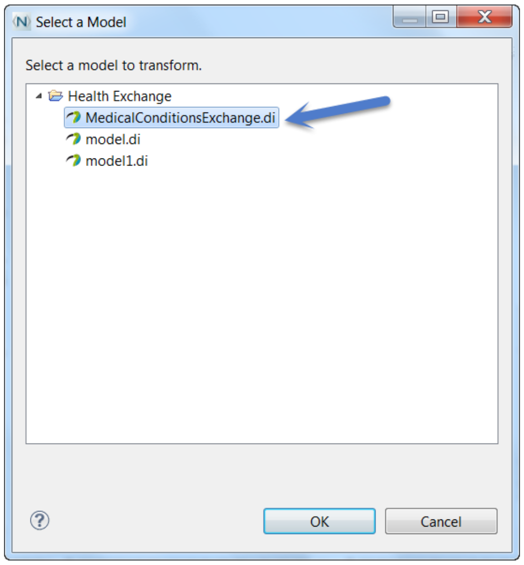
The property holder can be further refined but for a basic exchange we will use all properties in our schema . For a schema that supports multiple exchanges, creating multiple property holders can refine an exchange based on the action intended, e.g. create or update.

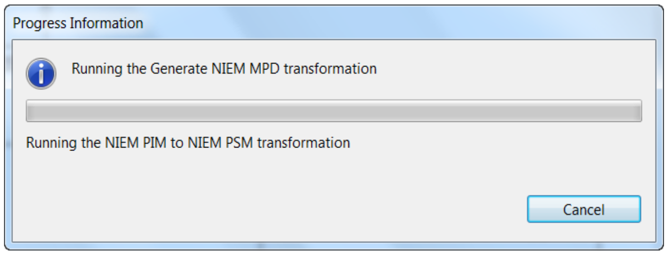
Generating the Model Description Package

And we are now ready to generate our Model Description Package.
From the NIEM UML tab in the menu bar, select the ‘Generate NIEM MPD’:


NIEM UML - Health Exchange/MedicalConditionsExchange.di - NIEM UML
File Edit Diagram Navigate Search Papyrus Project Run NIEM UML Window Help
Generate NIEM MPD Alt+Shift+U, G
Generate MPD Model Alt+Shift+U, A
Tahoma
Quick Access
Resource
NIEM UML
Papyrus
MedicalConditionsExchange.di
MedicalConditionsExchangeExchange
MedicalConditionsExchangeE...
MedicalConditionsExchangeE...
Palette
Nodes
Interface
InstanceSpecification
InformationItem
PrimitiveType
Model
Operation

We select our model and 'OK':

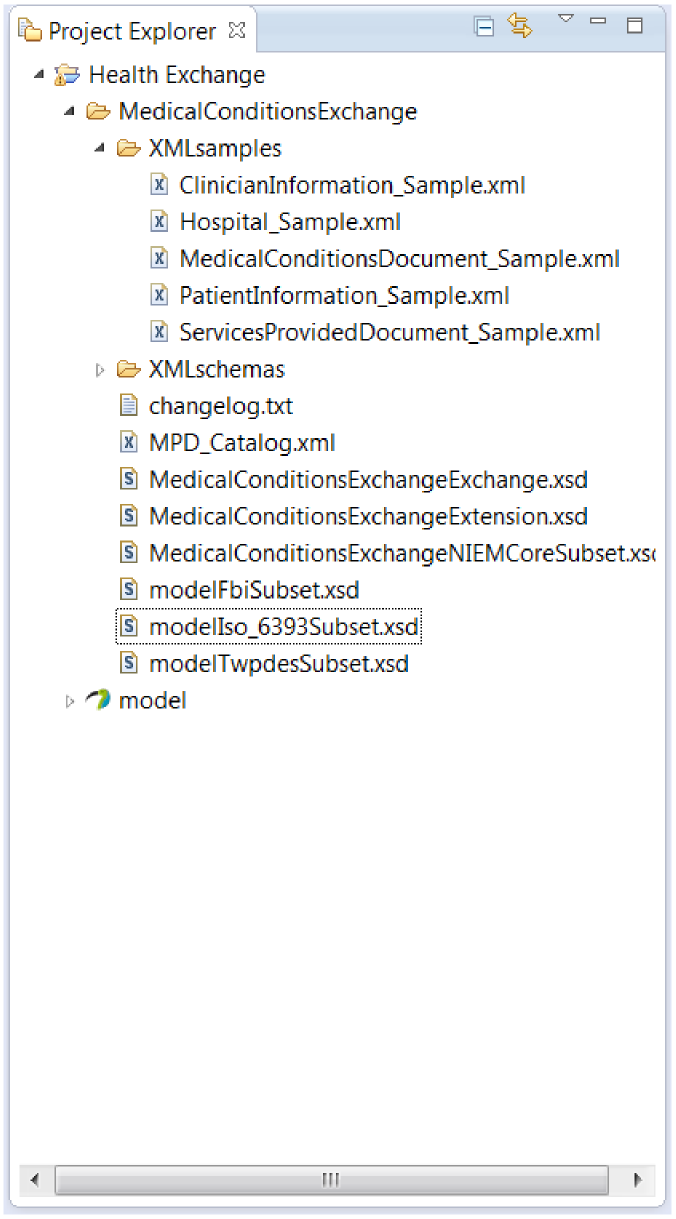



Progress Information

Running the Generate NIEM MPD transformation

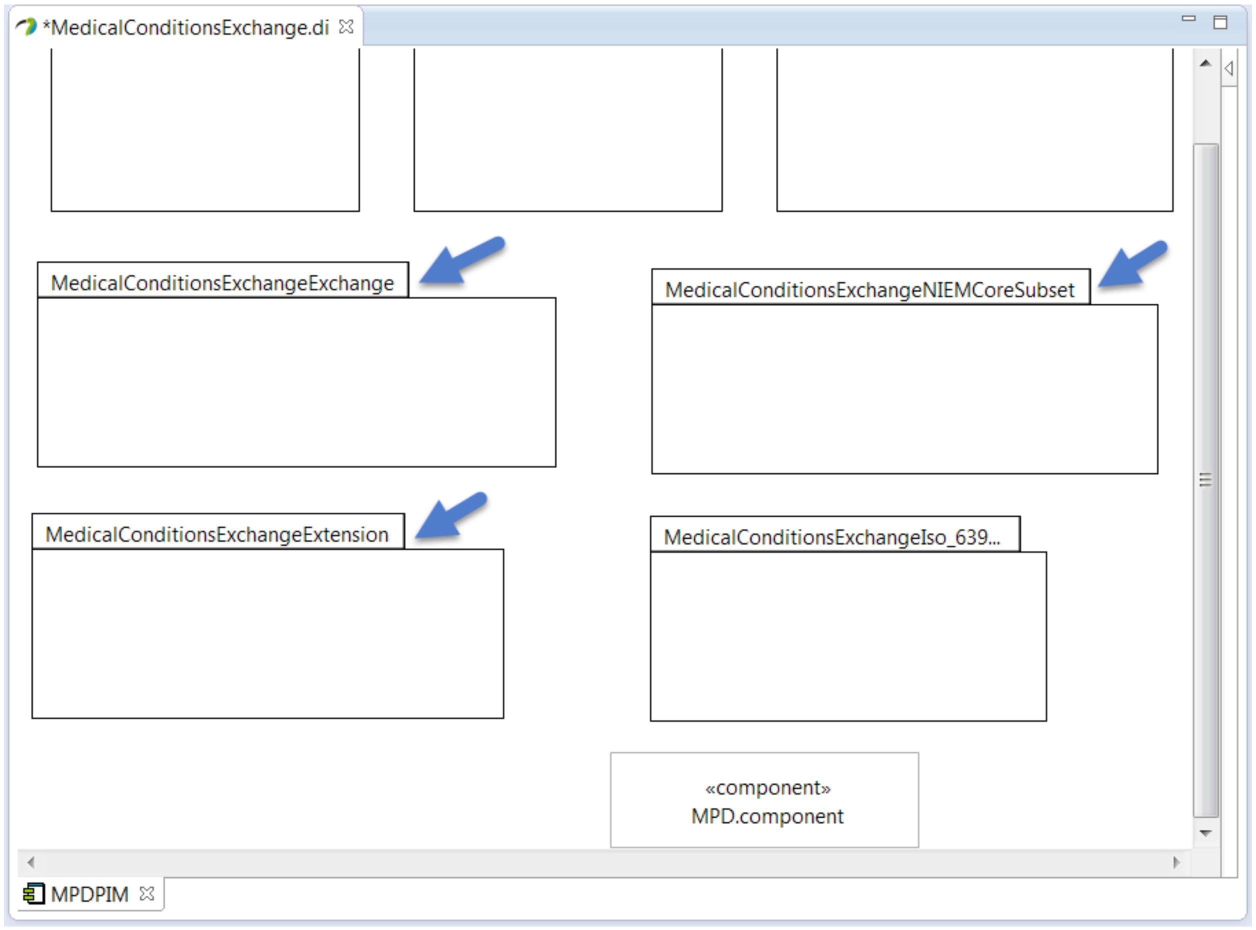
Running the NIEM PIM to NIEM PSM transformation

We have now transformed a NIEM Conformant MPD and stored in the workspace. The folders contents are depicted in the Project Explorer and can be reviewed through the tool.

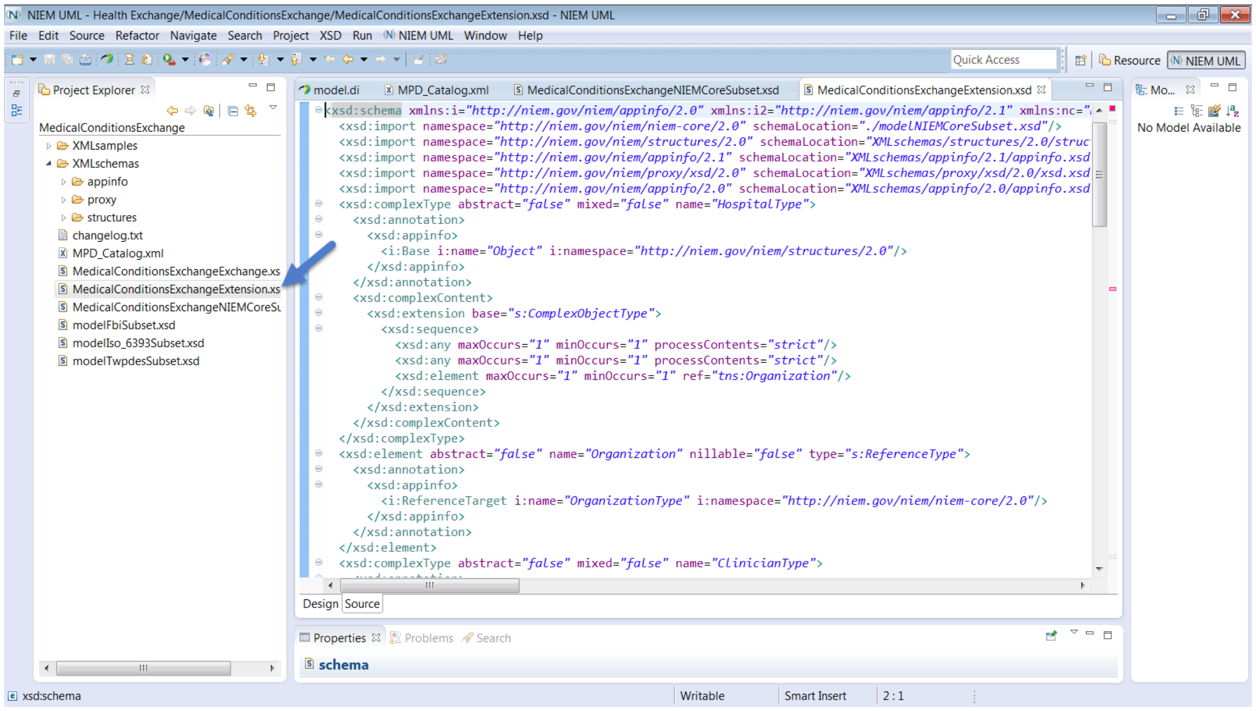


Health Exchange
MedicalConditionsExchange
XMLsamples
ClinicianInformation_Sample.xml
Hospital_Sample.xml
MedicalConditionsDocument_Sample.xml
PatientInformation_Sample.xml
ServicesProvidedDocument_Sample.xml
XMLschemas
changelog.txt
MPD_Catalog.xml
MedicalConditionsExchangeExchange.xsd
MedicalConditionsExchangeExtension.xsd
MedicalConditionsExchangeNIEMCoreSubset.xs
modelFbiSubset.xsd
modelIso_6393Subset.xsd
modelTwpdesSubset.xsd
model

Opening up the model we see the PIM packages and can navigate through the schemas and model diagrams, as before:


*MedicalConditionsExchange.di
MedicalConditionsExchangeExchange
MedicalConditionsExchangeNIEMCoreSubset
MedicalConditionsExchangeExtension
MedicalConditionsExchangeIso_639...
«component»
MPD.component
MPDPIM

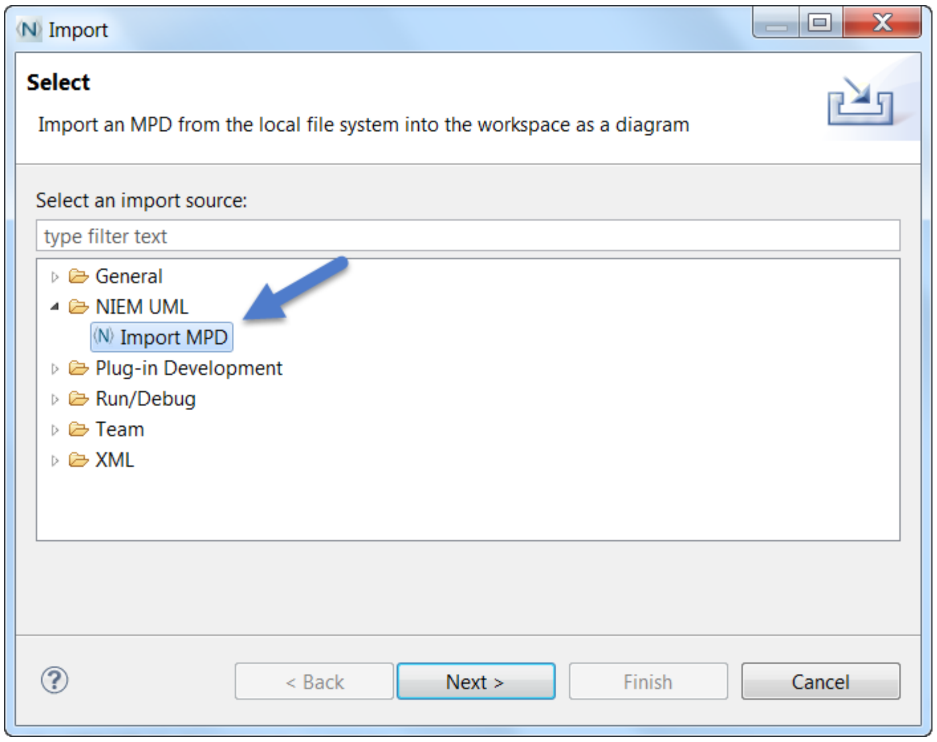
Double-clicking on the .xsd presents the generated XML, below is the XML generated for the Extension schema.


NIEM UML - Health Exchange/MedicalConditionsExchange/MedicalConditionsExchangeExtension.xsd - NIEM UML
File Edit Source Refactor Navigate Search Project XSD Run NIEM UML Window Help
Quick Access
Resource
NIEM UML
Project Explorer
MedicalConditionsExchange
XMLsamples
XMLschemas
appinfo
proxy
structures
changelog.txt
MPD_Catalog.xml
MedicalConditionsExchangeExchange.xs
MedicalConditionsExchangeExtension.xs
MedicalConditionsExchangeNIEMCoreSu
modelFbiSubset.xsd
modelIso_6393Subset.xsd
modelTwpdesSubset.xsd
model.di
MPD_Catalog.xml
MedicalConditionsExchangeNIEMCoreSubset.xsd
MedicalConditionsExchangeExtension.xsd
Mo...
No Model Available
Design
Source
Properties
Problems
Search
schema
xsd:schema
Writable
Smart Insert
2 : 1

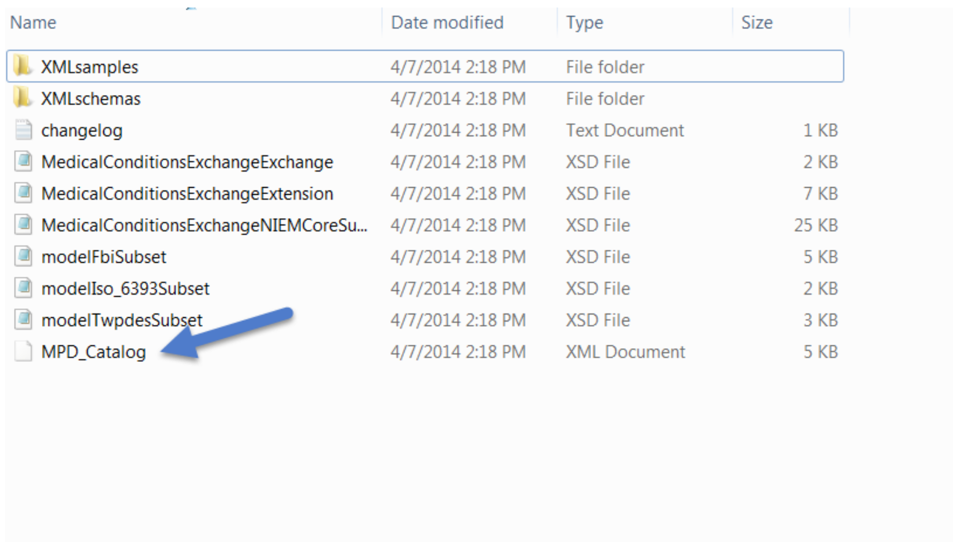
```
<xsd:schema xmlns:i="http://niem.gov/niem/appinfo/2.0" xmlns:i2="http://niem.gov/niem/appinfo/2.1" xmlns:nc="
  <xsd:import namespace="http://niem.gov/niem/niem-core/2.0" schemaLocation="./modelNIEMCoreSubset.xsd"/>
  <xsd:import namespace="http://niem.gov/niem/structures/2.0" schemaLocation="XMLschemas/structures/2.0/struc
  <xsd:import namespace="http://niem.gov/niem/appinfo/2.1" schemaLocation="XMLschemas/appinfo/2.1/appinfo.xsd
  <xsd:import namespace="http://niem.gov/niem/proxy/xsd/2.0" schemaLocation="XMLschemas/proxy/xsd/2.0/xsd.xsd
  <xsd:import namespace="http://niem.gov/niem/appinfo/2.0" schemaLocation="XMLschemas/appinfo/2.0/appinfo.xsd
  <xsd:complexType abstract="false" mixed="false" name="HospitalType">
    <xsd:annotation>
      <xsd:appinfo>
        <i:Base i:name="Object" i:namespace="http://niem.gov/niem/structures/2.0"/>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="s:ComplexObjectType">
        <xsd:sequence>
          <xsd:any maxOccurs="1" minOccurs="1" processContents="strict"/>
          <xsd:any maxOccurs="1" minOccurs="1" processContents="strict"/>
          <xsd:element maxOccurs="1" minOccurs="1" ref="tns:Organization"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element abstract="false" name="Organization" nillable="false" type="s:ReferenceType">
    <xsd:annotation>
      <xsd:appinfo>
        <i:ReferenceTarget i:name="OrganizationType" i:namespace="http://niem.gov/niem/niem-core/2.0"/>
      </xsd:appinfo>
    </xsd:annotation>
  </xsd:element>
  <xsd:complexType abstract="false" mixed="false" name="ClinicianType">
```

Importing an MPD

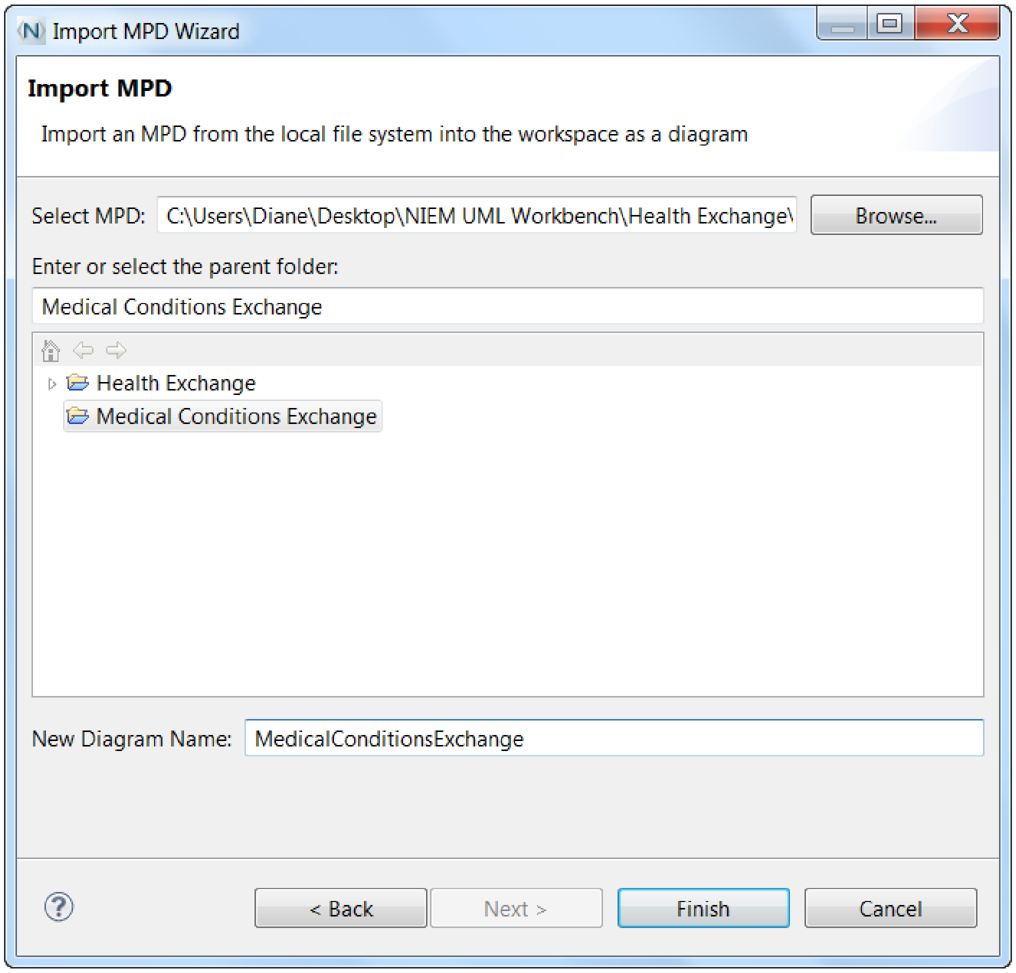
For demonstrations purposes, we are now going to close our model and import it back into the tool. We create a new 'Medical Conditions Exchange' project, and then from 'File' tab on the menu bar, click 'Import' and select the NIEM UML Import MPD.



Then 'browse' to the mpd.catalog file, give your new diagram a name and 'Finish'.



Name	Date modified	Type	Size
XMLsamples	4/7/2014 2:18 PM	File folder	
XMLschemas	4/7/2014 2:18 PM	File folder	
changelog	4/7/2014 2:18 PM	Text Document	1 KB
MedicalConditionsExchangeExchange	4/7/2014 2:18 PM	XSD File	2 KB
MedicalConditionsExchangeExtension	4/7/2014 2:18 PM	XSD File	7 KB
MedicalConditionsExchangeNIEMCoreSu...	4/7/2014 2:18 PM	XSD File	25 KB
modelFbiSubset	4/7/2014 2:18 PM	XSD File	5 KB
modelIso_6393Subset	4/7/2014 2:18 PM	XSD File	2 KB
modelTwpdesSubset	4/7/2014 2:18 PM	XSD File	3 KB
MPD_Catalog	4/7/2014 2:18 PM	XML Document	5 KB


Import MPD Wizard

Import MPD

Import an MPD from the local file system into the workspace as a diagram

Select MPD: C:\Users\Diane\Desktop\NIEM UML Workbench\Health Exchange\

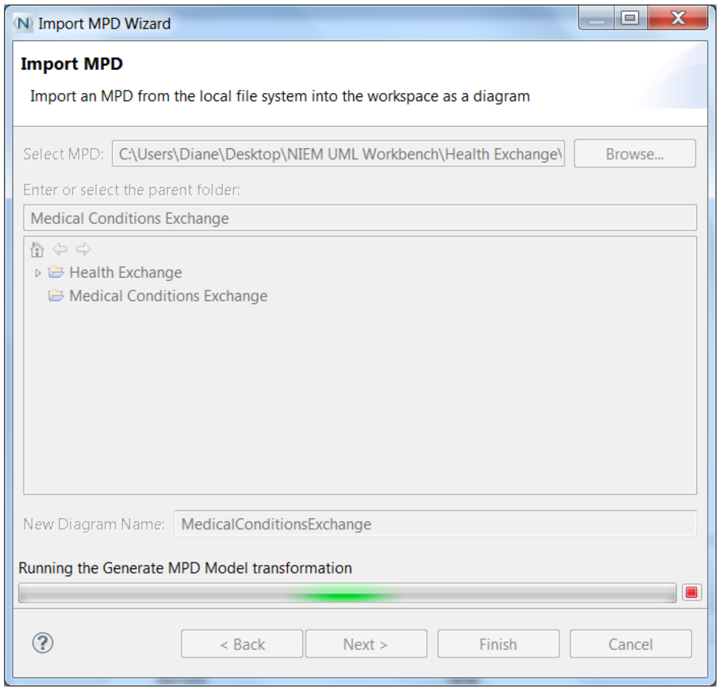
Enter or select the parent folder:

Medical Conditions Exchange

- Health Exchange
- Medical Conditions Exchange

New Diagram Name: MedicalConditionsExchange

In the same manner, you can import previously created IEPDs, as long as they have a conformant 'mpd.catalog' file.


Import MPD Wizard

Import MPD

Import an MPD from the local file system into the workspace as a diagram

Select MPD: C:\Users\Diane\Desktop\NIEM UML Workbench\Health Exchange\

Enter or select the parent folder:

Medical Conditions Exchange

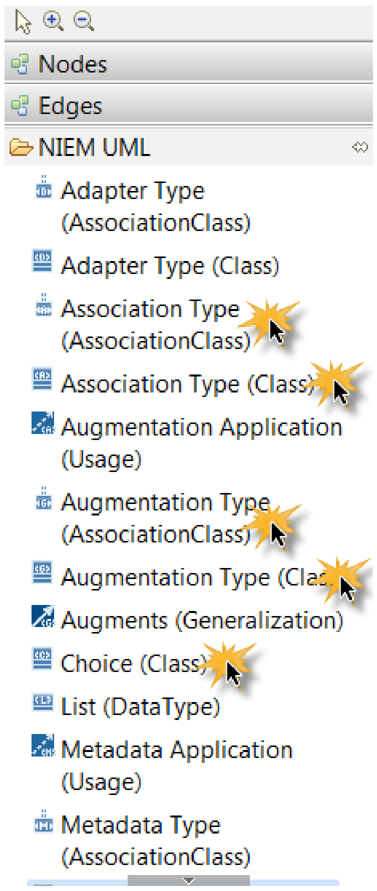
- Health Exchange
- Medical Conditions Exchange

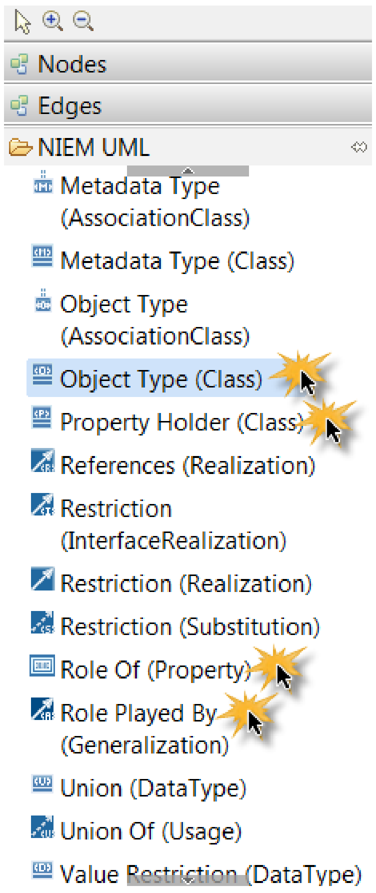
New Diagram Name: MedicalConditionsExchange

Running the Generate MPD Model transformation

NIEM Stereotypes

At this point, we’ve created our model, mapped our model to NIEM components, and generated our MPD. We’ve mostly used elements from the ‘Nodes’ and ‘Edges’ palette drawer. Now, we’re going to cover a few of the more commonly used stereotypes from the NIEM drawer, starred below. Note that the ‘NIEM UML’ drawer has ‘nodes’, ‘edges’, and ‘properties’ distinguishable by the icon. ‘Nodes’ are equivalent to ‘classes’, ‘edges’ equivalent to ‘associations’. The term ‘property’ is used in two definitions, both as a property or ‘attribute’ of a class and as a property of the attribute itself within the property window. We’ll cover the ‘starred’ classes, associations, and properties.


Nodes
Edges
NIEM UML
Adapter Type (AssociationClass)
Adapter Type (Class)
Association Type (AssociationClass)
Association Type (Class)
Augmentation Application (Usage)
Augmentation Type (AssociationClass)
Augmentation Type (Class)
Augments (Generalization)
Choice (Class)
List (DataType)
Metadata Application (Usage)
Metadata Type (AssociationClass)


Nodes
Edges
NIEM UML
Metadata Type (AssociationClass)
Metadata Type (Class)
Object Type (AssociationClass)
Object Type (Class)
Property Holder (Class)
References (Realization)
Restriction (InterfaceRealization)
Restriction (Realization)
Restriction (Substitution)
Role Of (Property)
Role Played By (Generalization)
Union (DataType)
Union Of (Usage)
Value Restriction (DataType)

Object Type (Class)

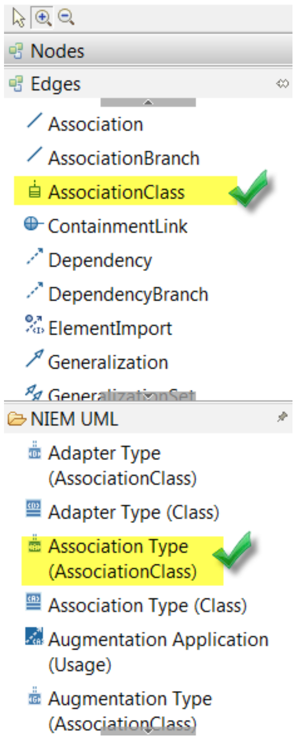
An object type is the default type for all UML classes. This type in the NIEM drawer is equivalent to the ‘class’ type in the ‘Nodes’ drawer and can be used interchangeably. Names of objects are not constrained as the mapping specification will map the UML names into NIEM conformant names. However it is good practice to use names that show some business intention – such as if the class is a person, type of person, e.g. customer, attorney, practitioner.

Property Holder (Class)

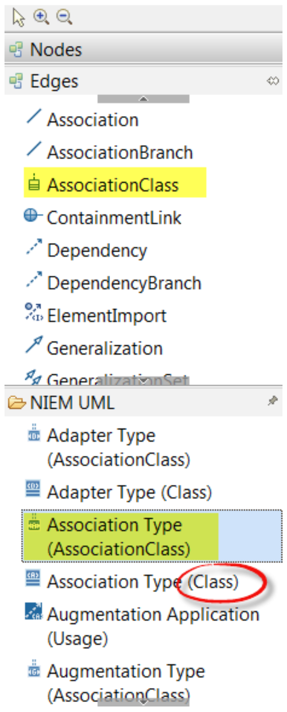
A property holder class is used to define a set of properties (attributes) that have no specific however or ‘top level’ properties. These properties are generally referenced by other properties. As noted earlier, this class is required in the exchange schema.

Association Type (AssociationClass) and Association Type (Class)

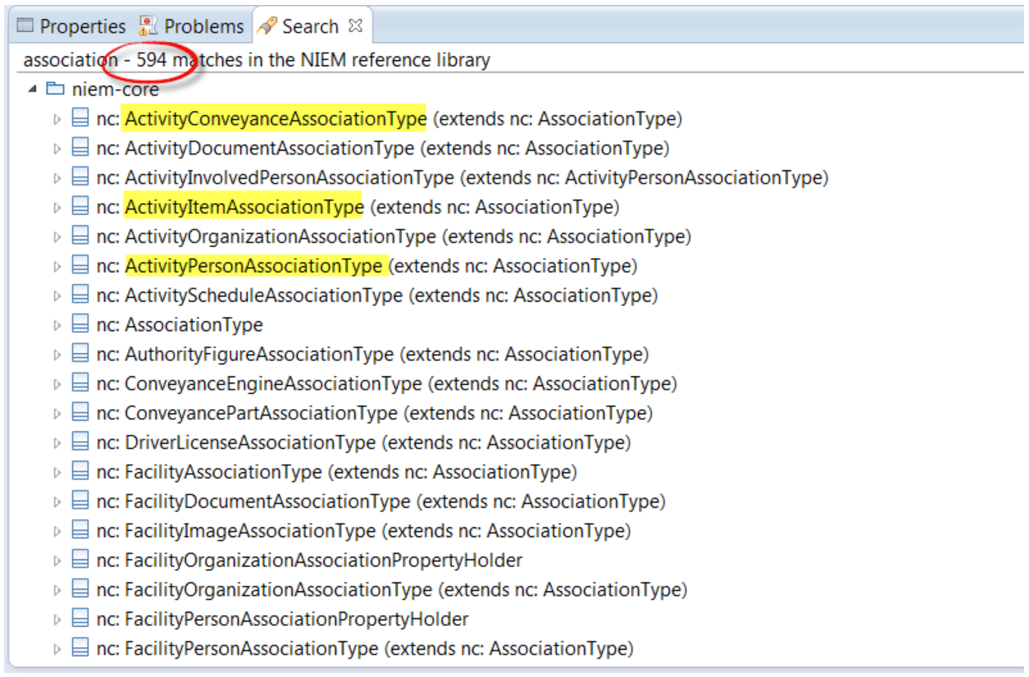
An NIEM Association Type (Association Class) is equivalent to a UML AssociationClass in the edge drawer and can be used interchangeably. A NIEM association is a specific relationship between NIEM objects used when a simple property is insufficient to model the relationship clearly. More specifically, is when properties of the relationship exist which are not attributable to the objects themselves. In our model, we could have defined 'Services Provided' as an association class between the clinician and the patient and created a direct relationship. One guideline to determine whether to create an association class or not is to consider the duration or 'life' of the association. For instance, a patient and a clinician are relatively permanent, whereas the services provided have a begin date and an end date. Another example might be a 'passenger' on a 'flight' with a 'ticket'. The passenger and flight would be modeled as objects, with the 'ticket' modeled as an association class.



The difference between 'Association Type (AssociationClass) and Association Type (Class) is that the connection relationships are presented in 'AssociationClass', whereas in 'Class' you are only presented with the class itself and can then create the connecting association.



NIEM model contains many association types, so before creating an extension association, thoroughly browse the NIEM reference library. The naming standards describe the type of association, e.g. an activity with a conveyance, activity with an item, an activity with a person.



association - 594 matches in the NIEM reference library

niem-core

- nc: ActivityConveyanceAssociationType (extends nc: AssociationType)
- nc: ActivityDocumentAssociationType (extends nc: AssociationType)
- nc: ActivityInvolvedPersonAssociationType (extends nc: ActivityPersonAssociationType)
- nc: ActivityItemAssociationType (extends nc: AssociationType)
- nc: ActivityOrganizationAssociationType (extends nc: AssociationType)
- nc: ActivityPersonAssociationType (extends nc: AssociationType)
- nc: ActivityScheduleAssociationType (extends nc: AssociationType)
- nc: AssociationType
- nc: AuthorityFigureAssociationType (extends nc: AssociationType)
- nc: ConveyanceEngineAssociationType (extends nc: AssociationType)
- nc: ConveyancePartAssociationType (extends nc: AssociationType)
- nc: DriverLicenseAssociationType (extends nc: AssociationType)
- nc: FacilityAssociationType (extends nc: AssociationType)
- nc: FacilityDocumentAssociationType (extends nc: AssociationType)
- nc: FacilityImageAssociationType (extends nc: AssociationType)
- nc: FacilityOrganizationAssociationPropertyHolder
- nc: FacilityOrganizationAssociationType (extends nc: AssociationType)
- nc: FacilityPersonAssociationPropertyHolder
- nc: FacilityPersonAssociationType (extends nc: AssociationType)

Augmentation Type (AssociationClass) and Augmentation Type (Class)

Augmentation is a NIEM specific type and only found in the NIEM drawer, but related to a ‘Generalization’ in the edge drawer. A NIEM augmentation type is a complex type that provides a reusable block of data that may be added to object type, or association types. Key aspects:

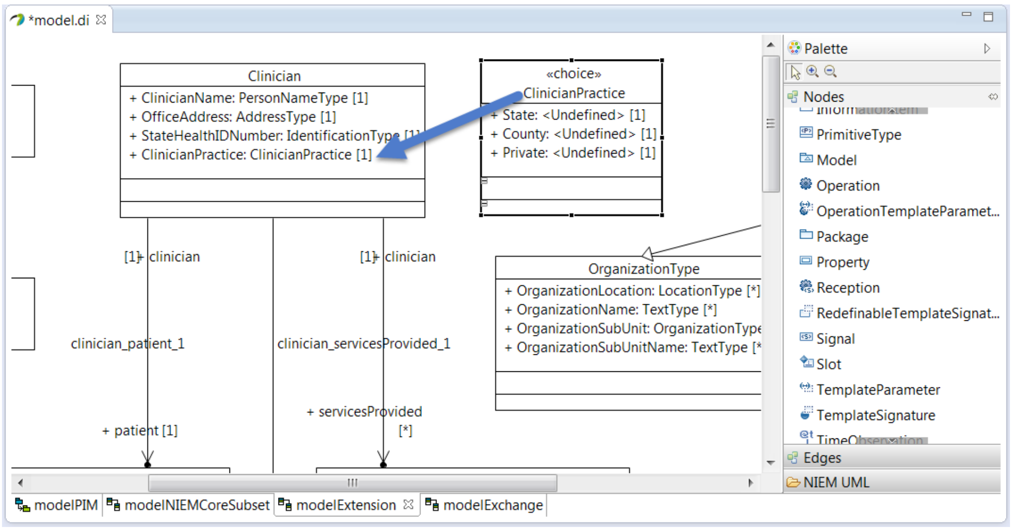
- A augmentation is structurally dependent upon other classes whereas an association class is not
- Augmentation is a NIEM type of UML generalization
- A NIEM Augmentation extends the properties specifically of class

As in associations, the difference between ‘Augmentation Type (AssociationClass) and Augmentation Type (Class) is that the connection relationships are presented in ‘AssociationClass’, whereas in ‘Class’ you are only presented with the class itself and can then create the connecting association.

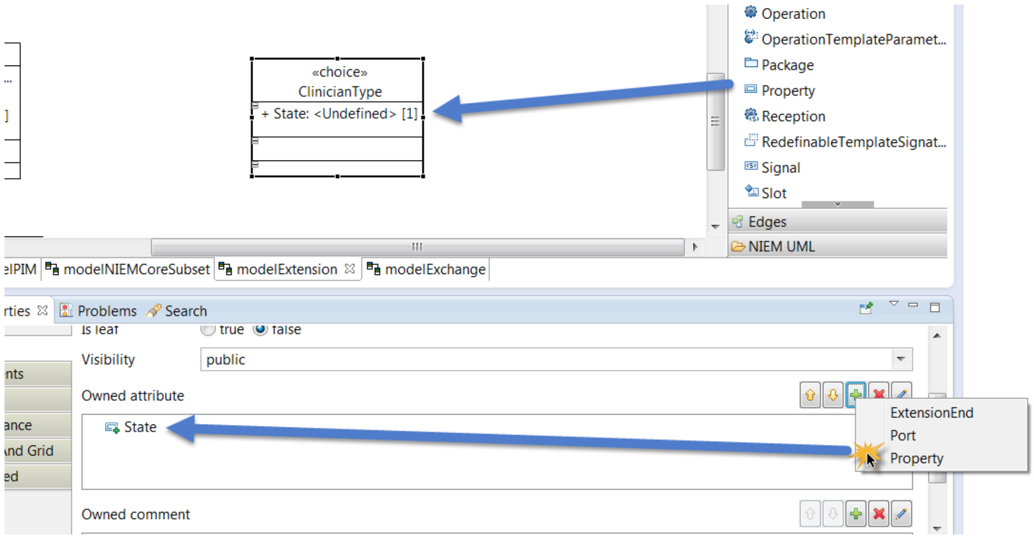
In practice, in augmentation is used to extend a NIEM type with properties that cannot be mapped to NIEM. The augmentation inherits all the properties of the NIEM type.

Choice (Class)

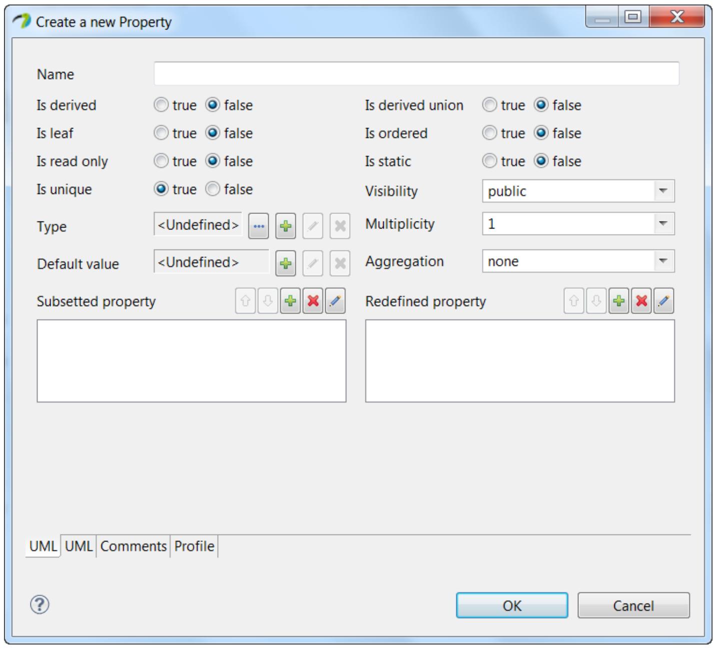
Choice is a NIEM type and only found in the NIEM drawer. A choice class is used to group a set of properties such that when used as the type of a property exactly one of them may have a value in any instance of an enclosing type. Below, we have created a ‘choice’ class from the NIEM UML drawer, then added a three properties to choose from, and associated that with the clinician class.


*model.di
Clinician
+ ClinicianName: PersonNameType [1]
+ OfficeAddress: AddressType [1]
+ StateHealthIDNumber: IdentificationType [1]
+ ClinicianPractice: ClinicianPractice [1]
«choice»
ClinicianPractice
+ State: <Undefined> [1]
+ County: <Undefined> [1]
+ Private: <Undefined> [1]
[1] clinician
[1] clinician
clinician_patient_1
clinician_servicesProvided_1
+ patient [1]
+ servicesProvided [*]
OrganizationType
+ OrganizationLocation: LocationType [*]
+ OrganizationName: TextType [*]
+ OrganizationSubUnit: OrganizationType
+ OrganizationSubUnitName: TextType [*
Palette
Nodes
PrimitiveType
Model
Operation
OperationTemplateParamet...
Package
Property
Reception
RedefinableTemplateSignat...
Signal
Slot
TemplateParameter
TemplateSignature
Edges
NIEM UML
modelPIM
modelNIEMCoreSubset
modelExtension
modelExchange

✓ Navigation Tip: You can also add a property through the property window under 'owned attribute',



which will bring up the 'Create a new Property' window:


Create a new Property

Name

Is derived	true [✓ false]	Is derived union	true [✓ false]
Is leaf	true [✓ false]	Is ordered	true [✓ false]
Is read only	true [✓ false]	Is static	true [✓ false]
Is unique	[✓ true] false	Visibility	public
Type	<Undefined>	Multiplicity	1
Default value	<Undefined>	Aggregation	none

Subsetted property

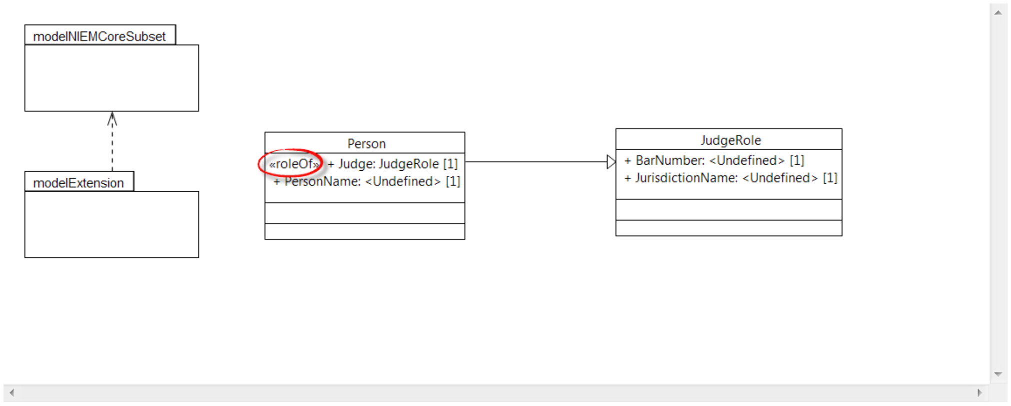
Redefined property

UML | UML | Comments | Profile

OK | Cancel

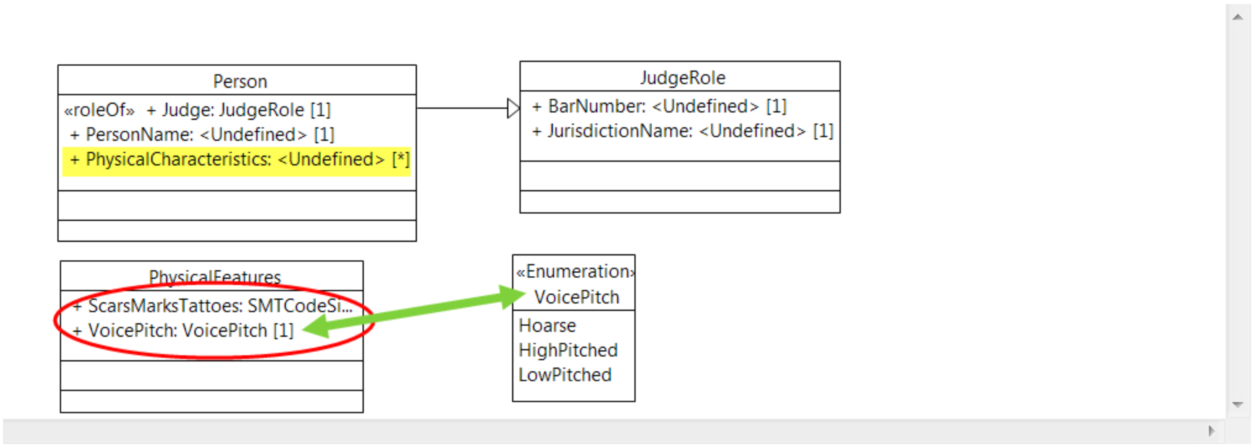
Role of (Property) and Role Played By (Generalization)

Roles are NIEM specific types. NIEM differentiates between an object and a role of the object. The term 'role' is used to mean a function or part played by some object. A class is interpreted as a role by means of a 'RoleOf' association end or a 'RolePlayedBy' generalization. A simple representation in an extension package is depicted below:

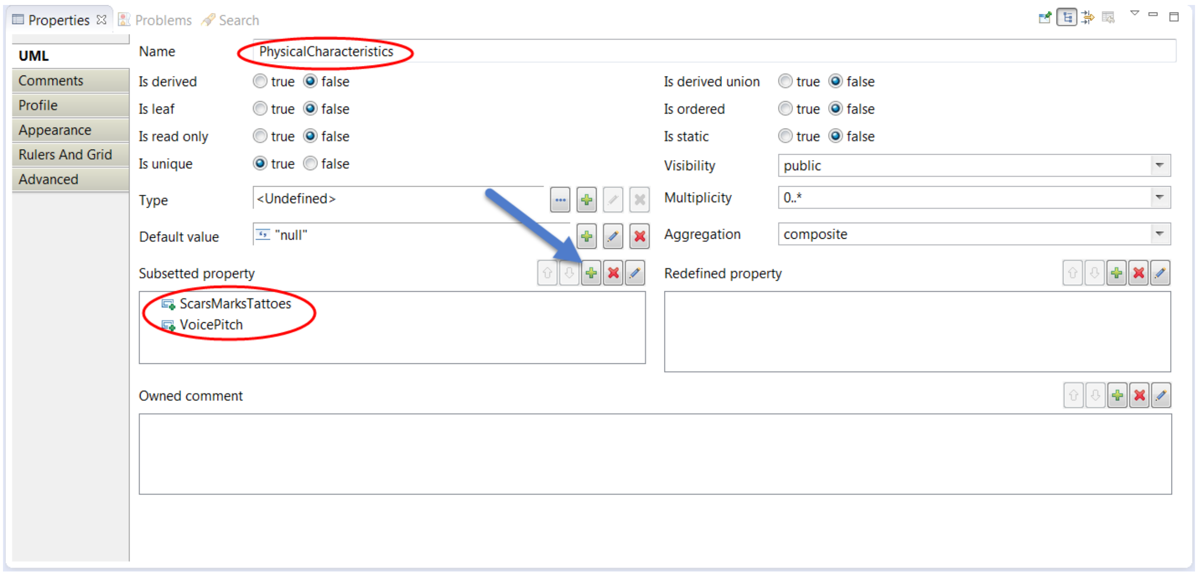

modelNIEMCoreSubset
modelExtension
Person
«roleOf» + Judge: JudgeRole [1]
+ PersonName: <Undefined> [1]
JudgeRole
+ BarNumber: <Undefined> [1]
+ JurisdictionName: <Undefined> [1]

Substitution Groups (Property)

UML Subsets express the NIEM Substitution Groups and are attained through the sub-setting property window. These define a property as being substitutable for another property. Again, a simple scenario is depicted below in the extension schema. Any physical characteristics want to be captured for a person and a 'PhysicalCharacteristics' property is created. A physical features class is created with a Scars, Marks, and Tattoes property which is mapped to the NIEM FBI Scars, Marks, and Tattoes code list. A 'VoicePitch' property is created and mapped to a 'VoiceP itch' enumeration.


Person
«roleOf» + Judge: JudgeRole [1]
+ PersonName: <Undefined> [1]
+ PhysicalCharacteristics: <Undefined> [*]
JudgeRole
+ BarNumber: <Undefined> [1]
+ JurisdictionName: <Undefined> [1]
PhysicalFeatures
+ ScarsMarksTattoes: SMTCodeSi...
+ VoicePitch: VoicePitch [1]
«Enumeration»
VoicePitch
Hoarse
HighPitched
LowPitched

Then, in the properties window of 'Physical Characteristics', 'ScarsMarksTattoes' and 'VoicePitch' are added to subsetted properties. Note that multiplicity is set to '0.*', providing unlimited physical characteristics from the two su bsetted properties.



Name	PhysicalCharacteristics		
Is derived	true [✓ false]	Is derived union	true [✓ false]
Is leaf	true [✓ false]	Is ordered	true [✓ false]
Is read only	true [✓ false]	Is static	true [✓ false]
Is unique	[✓ true] false	Visibility	public
Type	<Undefined>	Multiplicity	0..*
Default value	"null"	Aggregation	composite

Subsetted property

- ScarsMarksTattoes
- VoicePitch

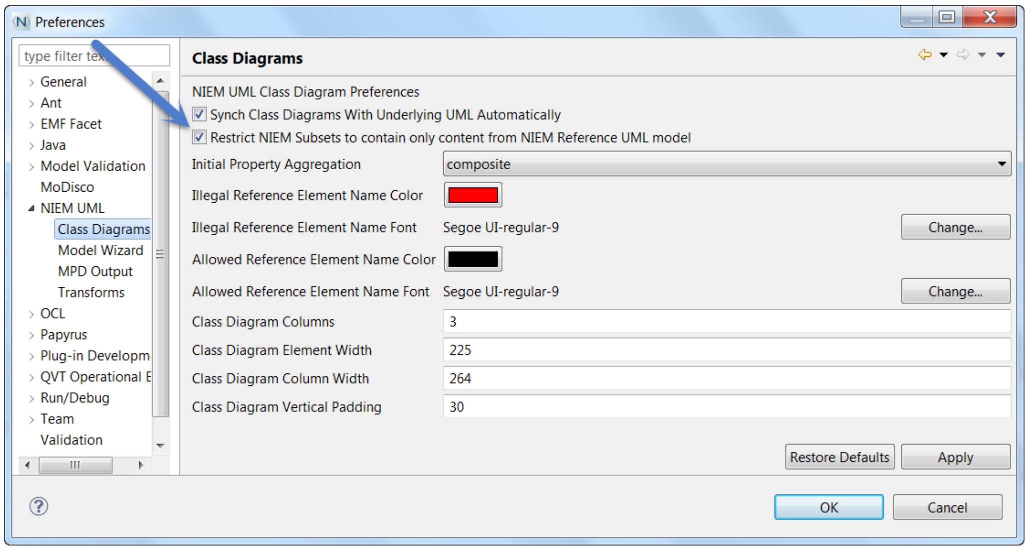
Redefined property

Owned comment

Tool Preferences

There are NIEM UML Preferences that can be set through the menu tab ‘Window > Preferences’ selection. The Class Diagram preferences are primarily presentation but have two significant check-boxes:

- ➢ The first checkbox presents or hides the underlying UML classes in the diagrams, the primitive types. Unchecking the box will hide the classes. **Note: This preference needs to be set prior to modeling in order to take effect**. On the one hand, unchecking the box will keep your diagrams less ‘busy’. On the other hand, you may be interested in know what is being included in your model.
- ➢ The second checkbox should remain checked; if unchecked, you run the risk of creating a non-conformant model.



Class Diagrams

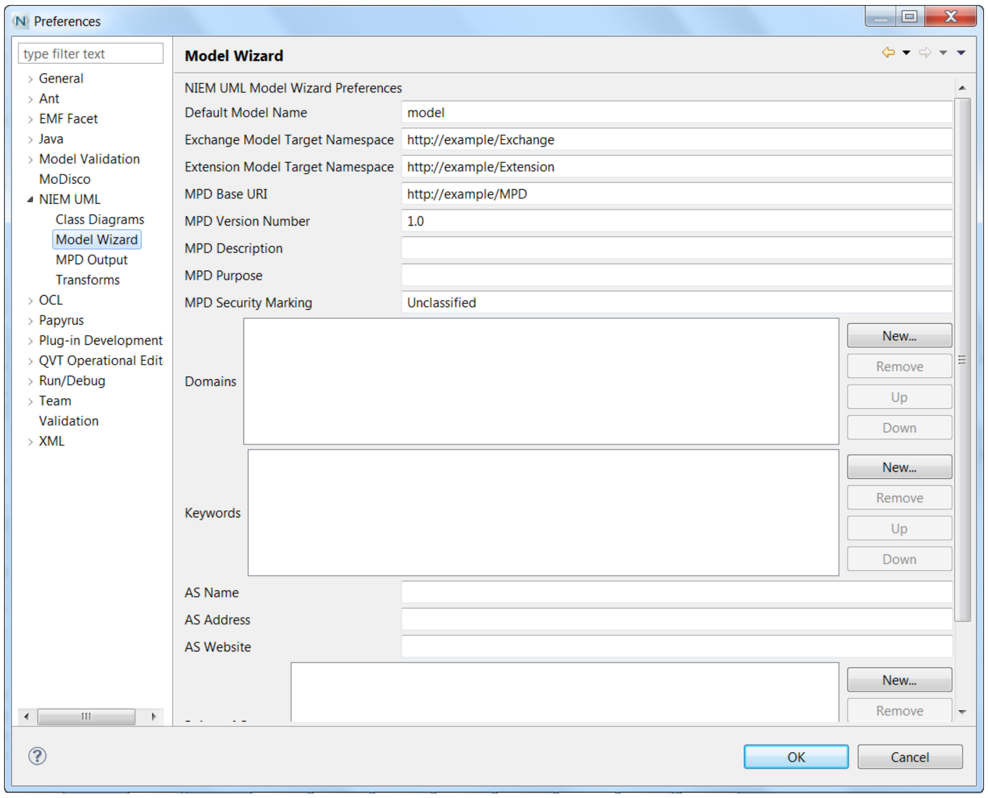
NIEM UML Class Diagram Preferences

☑ Synch Class Diagrams With Underlying UML Automatically

☑ Restrict NIEM Subsets to contain only content from NIEM Reference UML model

Initial Property Aggregation	composite
Illegal Reference Element Name Color	🟥
Illegal Reference Element Name Font	Segoe UI-regular-9
Allowed Reference Element Name Color	⬛
Allowed Reference Element Name Font	Segoe UI-regular-9
Class Diagram Columns	3
Class Diagram Element Width	225
Class Diagram Column Width	264
Class Diagram Vertical Padding	30

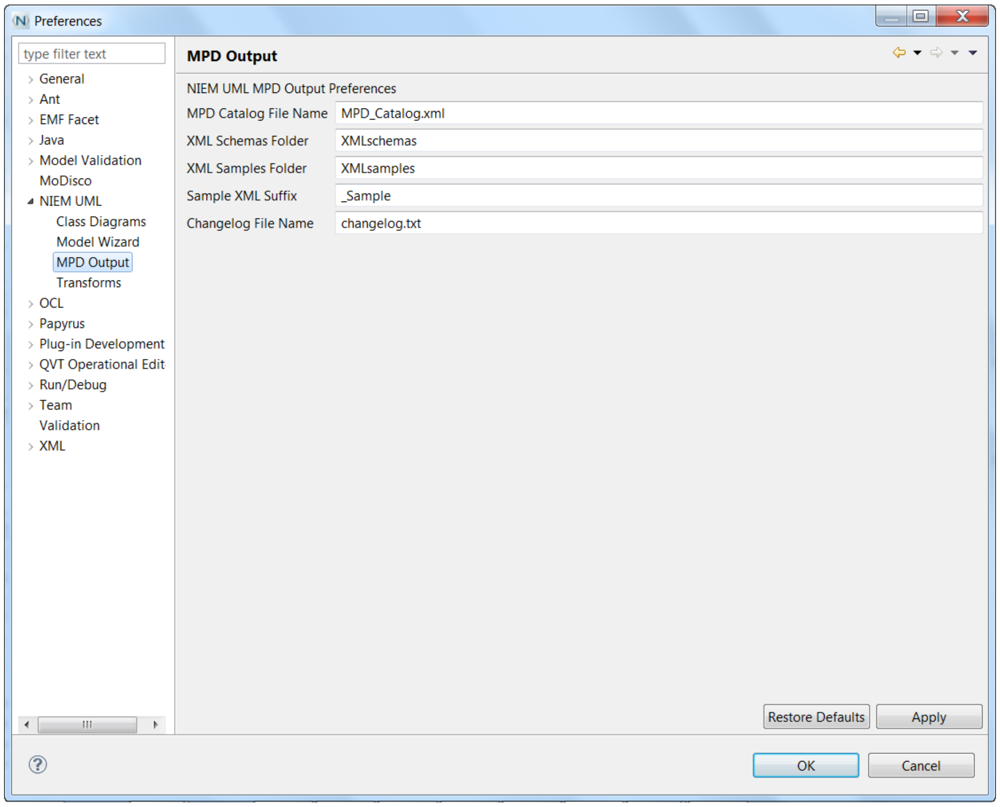
The Model Wizard and MPD Output preferences allow you to change the values for the MPD template and the default transformation outputs. At the moment, the tool only supports one template and set of defaults.



Model Wizard

NIEM UML Model Wizard Preferences

Default Model Name	model
Exchange Model Target Namespace	http://example/Exchange
Extension Model Target Namespace	http://example/Extension
MPD Base URI	http://example/MPD
MPD Version Number	1.0
MPD Description	
MPD Purpose	
MPD Security Marking	Unclassified
Domains	
Keywords	
AS Name	
AS Address	
AS Website	



MPD Output

NIEM UML MPD Output Preferences

MPD Catalog File Name	MPD_Catalog.xml
XML Schemas Folder	XMLschemas
XML Samples Folder	XMLsamples
Sample XML Suffix	_Sample
Changelog File Name	changelog.txt

Further tutorial materials will cover development of more complex exchanges and NIEM UML Stereotypes.

This concludes the Open Source NIEM UML Toll - Basic Exchange Tutorial.