Enterprise Architect

User Guide Series

XML Schema (XSD)

In Sparx Systems Enterprise Architect, you can use structural models to define the meta-model of a domain and generate XSD or WSDL schema. Use the Schema Composer to build or import complex XSD or WSDL, with support for MOF, ODM and the NIEM core.

Author: Sparx Systems
Date: 2026-05-28
Version: 17.1

CREATED WITH ENTERPRISE ARCHITECT

Table of Contents

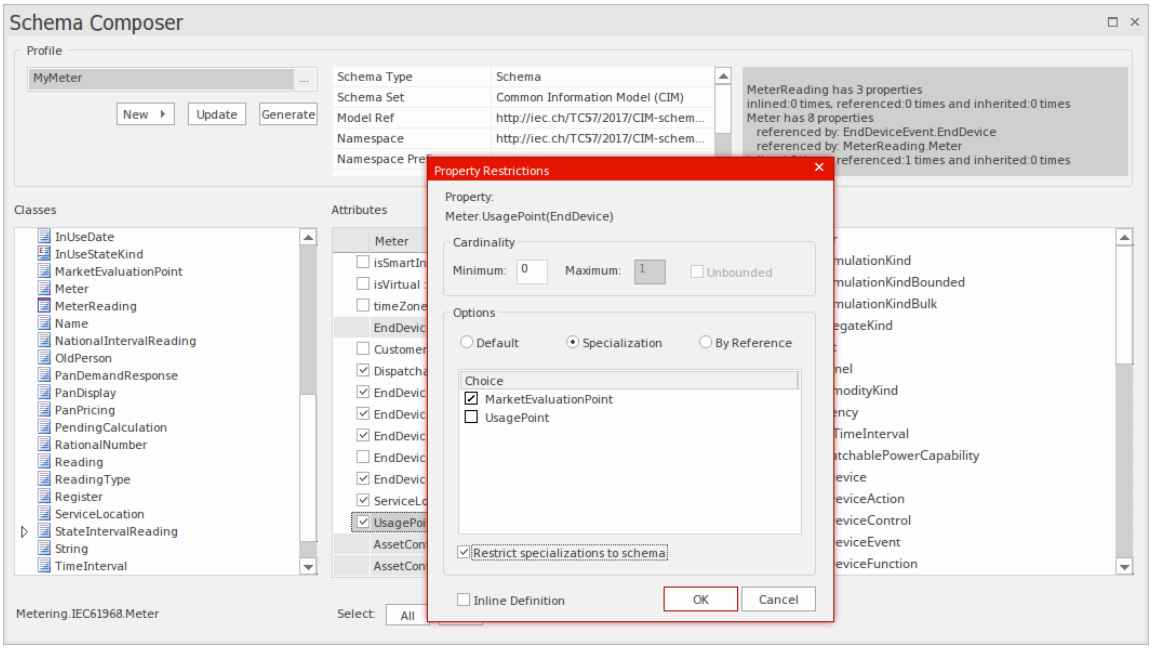
XML Schema (XSD) 6
The Schema Composer 9
Schema Composer Profiles 14
Create a Schema Profile 18
Schema Compositions 22
Class Diagrams 31
Schema Analysis 34
Generate Schema 35
Select a Schema Profile 37
Generate Schema File 39
CIM Schema Guide 42
NIEM Schema Guide 44
UPCC Schema Guide 46
Model Compositions 48
Generate a Model Subset (Transform) 51
UML Profile for Core Components (UPCC) 56
Available Frameworks 62
Install a Core Framework 67
The Schema Importer 71
Schema Composer Automation Integration 75
Schema Composer Addin Integration 76
Schema Composer Scripting Integration 77
MDG Technologies - UML Profile Extensions 86
XSD Models 89

Modeling XSD 90
XSD Diagrams 94
Schema Package 95
Global Element 99
Local Element 103
Global Attribute 108
Local Attribute 112
Attribute Group 116
Complex Type 119
Simple Type 123
Group 127
Any 131
Any Attribute 135
Union 139
Model Group 144
Enumeration 149
XML from Abstract Class Models 153
Default UML to XSD Mappings 158
Generate XSD 162
Generate Global Element 166
Import XSD 168
Global Element and ComplexType 172
XSL Transforms 174
Model an XSL Transformation 179
Execute an XSL Transformation 184
Debug an XSL Transformation 186
XML Validation 188

XML Service Oriented Architecture 194
WSDL 197
WSDL 1.1 Model Structure 198
Model WSDL 202
WSDL Namespace 207
WSDL Message 210
WSDL Message Part 213
WSDL Port Type 217
WSDL Port Type Operation 220
WSDL Binding 226
WSDL Binding Operation 230
WSDL Service 236
WSDL Document 240
Generate WSDL 244
Import WSDL 248
SoaML 251
SoaML Toolbox Pages 255
SOMF 2.1 258
National Information Exchange Modeling (NIEM) 2.1 260
National Information Exchange Modeling (NIEM) 271
UML Profile for NIEM 273
Download the NIEM Reference Model 292
Creating a NIEM IEPD 294
Customize Your IEPD Model 301
NIEM IEPD Generation 307
Creating a NIEM Data Model 310
Subsetting NIEM with the Schema Composer 316

Walk Through Examples 323
Example NIEM Schema 335
Import NIEM XML Schema 348

XML Schema (XSD)



Structural models in Enterprise Architect, especially Class models, are frequently used to define the meta-model of some domain of interest. For example a meta-model can be defined using a Class model to rigidly define the objects, data, relationships and types that make up the domain of Geospatial information. Likewise, models can be (and are) built to describe domains such as Water Management, Health, Retail, Insurance, Car Registration, Entertainment and many more.

These models are extremely valuable and frequently represent a significant investment in time and money by either commercial or standards based organizations. An important part of realizing the benefit of these models, in particular where information must be exchanged between multiple parties, is in the definition of schema (often XSD based) that codify how a message should be formed to be

conformant to the underlying meta-model. Traditionally, such message schema are written by hand, based on the meta-model. This is generally a laborious and error prone exercise.

Enterprise Architect has a long history of being associated with the development of both commercial and standards based meta-models, and there are many examples of models defined in Enterprise Architect model files that are used to specify the exact construction of an information domain of interest.

The Schema Composer in Enterprise Architect has been built to take maximum advantage of models stored in an Enterprise Architect model file or repository (or Cloud based server) by streamlining the conversion of model information into schemas that comply with the naming standards and format of a variety of popular industry meta-models. This approach drastically reduces the time taken to form a valid schema and eliminates human error in transcribing model information into schema text.

The current version of the Schema Composer supports XSD generation for a number of technologies, and in addition supports the customization of output by integrating tightly with both the Automation Interface and the Add-In framework. In this manner it is possible to use one of the schema generators supplied 'As-Is' or to write a custom generator using JavaScript, or to go further and fully customize the process by writing a suitable Add-In in a language of choice.

In addition to the new Schema Composer, Enterprise

Architect also supports the modeling of XSD and WSDL definitions using UML Profiles that support explicit modeling of the relevant types. This is sometimes necessary when building a complex XSD or WSDL from scratch and needing to have a fully worked out visual model of the final schema. Note that as Enterprise Architect also supports the import of XSD documents, it is possible to produce a schema using the Schema Composer, and then for documentation and visualization purposes (or even for further customization), import that schema back into either the current or a different model.

Additional topics included in the Schema Engineering section are devoted to the Meta Object Facility (MOF), the Ontology Definition Metamodel (ODM) and the National Information Exchange Model (NIEM). The section on NIEM is quite extensive, as Enterprise Architect includes many features necessary to model and work with NIEM domains and schema. As with some of the other technologies, there is in addition a downloadable version of the NIEM core as an Enterprise Architect model.

The Schema Composer

Seamlessly Model Schema Compliant Message Definitions in a Simple and Productive Tool

The Schema Composer is a versatile tool for quickly and easily defining a variety of formal schema from a model. Due to the unique nature of the Schema Composer, it is not necessary to use a profile or stereotyped elements when building the definition of an XSD (or other) document. This greatly enhances the re-usability of the underlying model and helps alleviate the complexity that arises when dealing directly with XSD or other element types and restrictions.

Many industries have worked hard over the last decade to define shared meta-models specific to their industry, and it is these models that now form the basis for contractual information sharing across organizations and across geographic borders. A typical usage scenario of the Schema Composer is in the creation of message definitions (schema) to exchange information between organizations, ensuring that such messages comply with the underlying meta-model that has been adopted by the involved parties.

When information is shared between organizations, it is frequently the case that only a subset of the full meta-model is required, but it is essential that what is shared conforms precisely to the agreed meta-model. In this case the Schema Composer is the perfect tool for deriving contractual schema based on sub-sets and restricted data sets that take a 'slice'

through the meta-model as a whole.

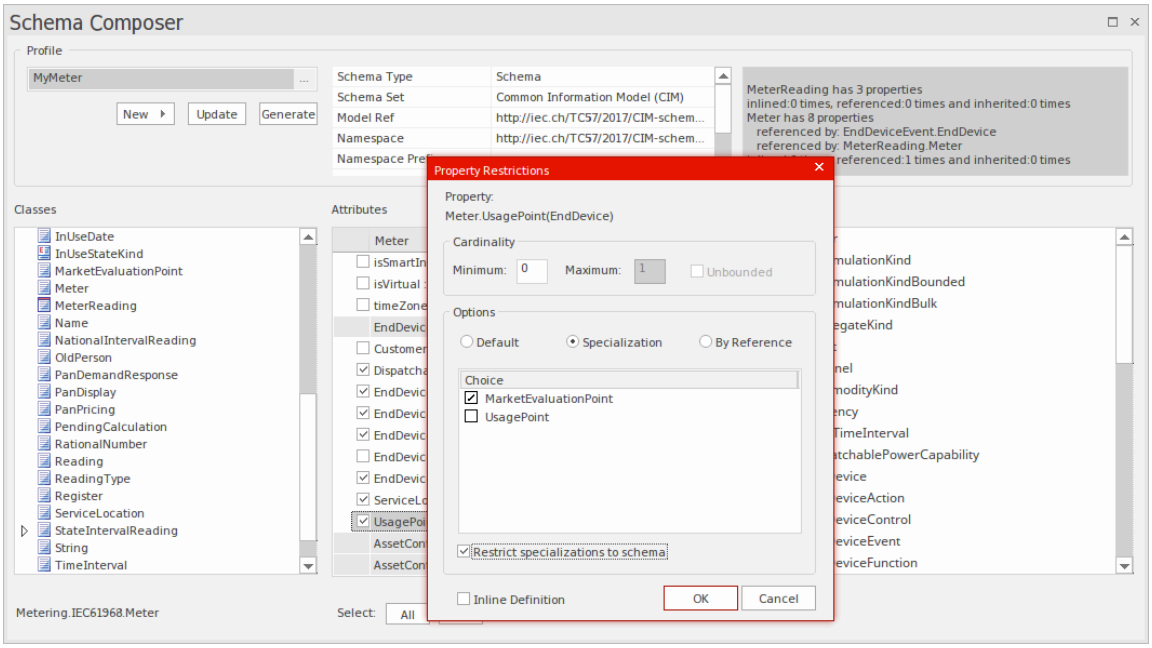
The Schema Composer avoids the common 'pain points' of working with XSD and other schema languages directly:

- There is no need to create a relatively complex XSD model composed of specific XSD elements, in addition to your 'normal' business and data models, to define the required data, its associations and references, and any restrictions or conditions
- You do not need to understand how to use the XSD elements and apply the XSD naming rules and conventions to correctly construct such models; formatting and naming rules as specified by the supported standards are automatically taken care of

The Schema Composer greatly simplifies the process of creating standards compliant schema in a re-usable and accessible manner. In this illustration, you can see how a simple Class diagram is used as the source for the Schema Composer to generate XML Schema.

The Schema Composer is supported in the Corporate, Unified and Ultimate Editions of Enterprise Architect

Schema Composer



This figure shows a Schema Composition for the Process Order domain in the Example model.

Access

Ribbon	Develop > Schema Modeling > Schema Composer > Open Schema Composer

Benefits

The Schema Composer:

- Operates on a Class model rather than an XML schema profile
- Relieves you of the XSD-specific design and schema generation decisions, whilst still ensuring consistency

across the profile

- Can operate on a generic Class model to provide generic XSD documents
- Is most useful when operating on industry standard Class models that have specific domain based meaning
- In most circumstances operates on a full model from which a subset of properties from selected Classes are drawn to build specific messages, to communicate only what is necessary for the information to send or request
- For standards such as NIEM, will generate a new sub-model as part of a broader NIEM compliant schema definition

Standards that the Schema Composer currently supports include:

- The Common Information Model (CIM)
- National Information Exchange Modeling (NIEM)
- United Nations Center for Trade Facilitation and Electronic Business (UN/CEFACT) Modeling Methodology (UMM), specifically the Naming and Design Rules (NDR) 2.1 and 3.0
- Universal Business Language (UBL), specifically the Naming and Design Rules (NDR) 3.0

The Schema Composer also helps you to build a definition of the same message using different formats such as:

- XSD
- RDFS
- JSON

In addition the Schema Composer:

- Supports formats implemented using a custom Add-In that takes advantage of the Schema Composer automation interface
- Has built-in support for various serialization formats and styles used by different industry models

Schema Composer Profiles

Schema Composer profiles are the configuration files that describe the elements and restrictions that will make up a particular schema or sub-model. Profiles are generally tied to a particular technology such as the Common Information Model (CIM) or the UML Profile for Core Components (UPCC), and the interpretation of the material within the Profile and the nature of the schema or sub-model published will be dependent on the technology-specific generator used. While Enterprise Architect supports a number of technologies 'out of the box' (and more are planned), it is also possible to customize the process by taking advantage of the extensive automation interface in Enterprise Architect to leverage the rich content of Schema Composer Profiles under your own terms, either in an Add-In or script.

Schema Profiles

A Schema Composer profile comes in two forms. Each form fulfills a particular system requirement - Schema generation (xsd, rdfs, json) and sub model creation. When you create a profile in the Schema Composer you choose which form to use based on your needs. A single profile in the Schema Composer can be used to either compose a schema, *in its common forms,* or create a UML sub-model from a core model.

Profile Types

Type	Description
Model Transform	A profile of this type is used to generate a sub-model from a core model.
Schema	A profile of this type is used to generate a schema; typically an XSD schema representing messages, but also other formats such as JSON object notation and resource descriptor formats.

Schema Composition Methodologies

National Information Exchange Model (NIEM)

Enterprise Architect provides a NIEM framework and Schema Composer for generation of sub-model and XML schemas.

Common Information Model (CIM)

Enterprise Architect Schema Composer supports provides the CIM standard out of the box, for composition of CIM

compliant schema.

Universal Business Language (UBL)

Enterprise Architect provides a Universal Business Language framework, and the Schema Composer which provides the UBL standard for schema generation.

Core Component Technical Specification (CCTS) UN/CEFACT

Enterprise Architect provides a UML Profile for Core Components framework and Schema Composer. The Composer can generate business components libraries from core component libraries and simplifies the composition / publication of schema from message assemblies / business information entities.

Generic

Where a standard does not meet your requirements, the generic option provides a simpler choice for quick schema composition from your UML model. Typically you will model your own data library using UML Classes with attributes, associations, Aggregation and Inheritance. You can then use this model as the input to the Schema Composer.

EA Script Engine

Enterprise Architect provides a scripting engine that supports JavaScript, VBScript and JScript languages. The scripting engine is also integrated with the Schema Composer. When generating a schema, either for a particular standard or generic scheme, a script can be employed to perform the operation on its own or as a supplement to the options provided by the standard.

EA Add-In

Enterprise Architect provides Add-In integration with the Schema Composer. An Add-In can participate in the generation of the sub model or schema by registering its interest with Enterprise Architect. The Add-In can provide options and alternatives to be listed in the 'Schema Generation' dialog, and will be invoked should its options be chosen. The Add-In can access the content of the profile using the Schema Composer automation interfaces.

Create a Schema Profile

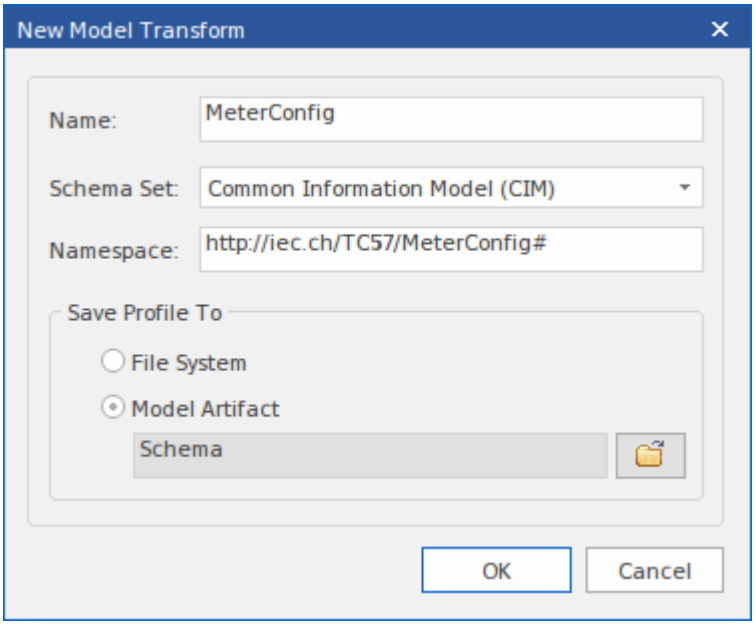
A schema profile identifies the name, technology and content of the schema as a precursor to defining how the schema is generated. You can create and edit as many schema profiles as you need. Schema profiles are bound to a single technology and will either map to a generated schema or a sub-setting transform.

Access

Ribbon	Develop > Schema Modeling > Schema Composer > Open Schema Composer

Creating a new Profile

If you are creating a schema for a particular technology, start by opening a model that has the required meta-model loaded. Sparx Systems make a number of meta-models available when using the Model Builder and/or the Sparx RAS/Cloud services. Follow these steps to build a new Profile. With the Schema Composer displayed, click on the 'New' button and select the profile type, either Schema or Transform.



The New Profile Screen

Option	Action
Schema Set	Select the standard to use, or choose the 'Generic' option.
Namespace	Depending on the standard you have selected this field might take an automatic value or remain blank. Provide a relevant namespace if blank. Refer to the next section for a description of how namespaces are managed in the Schema Composer.
Save profile To:	Profiles can be stored in the file system or in the model. Profiles stored in the model

	can be shared with others, while file system profiles are private.
OK	Click on this button to edit the new schema in the Composer.

Namespaces

When a new profile is created, you specify the target namespace and the namespace prefix. Schemas typically involve multiple namespaces and the Schema Composer provides support for this within a single model. The scheme by which namespaces are identified is the presence of two specific properties on a Package. The properties are 'URI', which specifies the namespace, and 'Alias', which provides the namespace prefix. The properties can be present on the immediate Package or a parent Package. When present, elements of the Class will take that namespace. Where no namespace exists, Classes will take the target namespace specified when the profile was created.

Save the profile

Click the Update button to save the profile you have just created.

Notes

- The process of creating and generating schema for NIEM has additional notes in the *NIEM* Help topic
- The Schema Composer is supported in the Corporate, Unified and Ultimate Editions of Enterprise Architect

Schema Compositions

A schema composition refers to a restricted set of elements taken from the model that together describe a unique entity that has no equivalent in the model. Commonly, schema compositions are used to generate schema files such as XSD files. In contrast, model compositions are used to configure the material as the basis of a subset 'transform' - for example when creating a NIEM model subset.

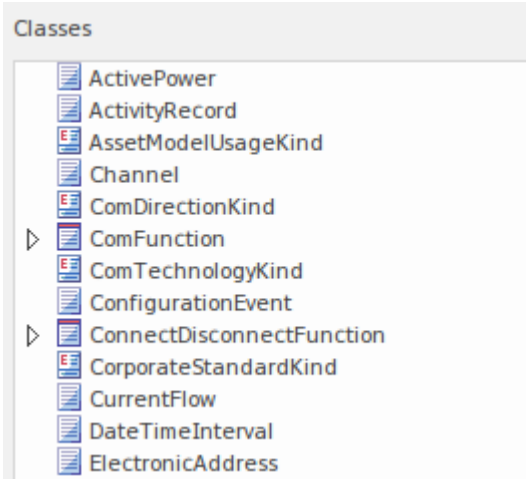
Define Schema Content

These steps walk you through the basic procedure of composing types in a Schema profile and show how you can restrict the content of elements to meet the message requirements.

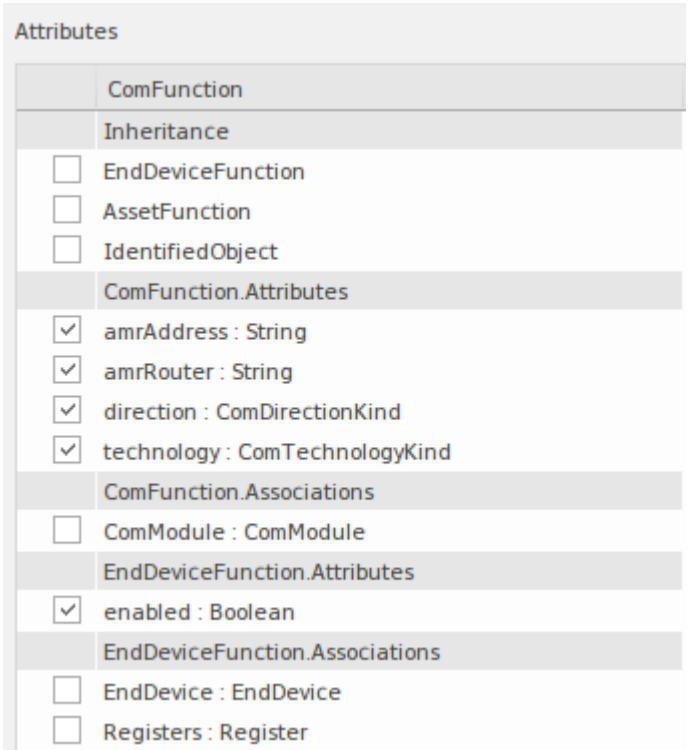
Add Classes

Drag the required Class elements from the Browser window into the 'Classes' panel. As you add a Class:

- Its ancestry is listed in the 'Inheritance' section in the middle panel
- Its attributes are listed under the 'Inheritance' section, with a blank checkbox against each one; Association and Aggregation entries are named according to the role name on the connector
- Its model structure path is shown underneath the 'Classes' panel



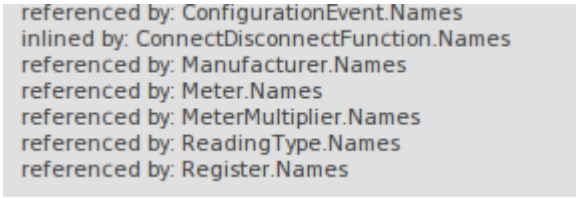
Select Properties



Any time you select a Class in the 'Classes' list, its attributes and model ancestry are listed in the 'Attributes' list. Select the checkbox against each attribute to define the elements of this type. When chosen, the attribute's type is added automatically to the schema, appearing in the 'Classes' list and the 'Schema' panel to the right.

When an attribute is unchecked, the type is not

automatically removed. Types can be removed using the Class context menu. It is worth noting that each time a Class is selected, all references to the Class are displayed in the status panel, enabling you to quickly review any Class usage.



Inheritance

If you favor or foresee a need for inheritance in the schema you are preparing, it would make sense to begin the composition with ancestors first, then re-use these as child Classes are added. The method is not set in stone. You can switch from an inheritance model to an aggregated composition or vice-versa at any time. Here is a brief description of the provision of inheritance in the Schema Composer.

The Schema Composer offers flexibility in dealing with inheritance. For example, you can choose to aggregate selected attributes from the Class and its parent, while choosing to inherit the grandparent. However, when you choose to use inheritance, you choose to inherit the restricted form of that type as well. When an ancestor is selected in this list, the generated XML schema would show an extension element identifying this ancestor. Only one ancestor can be selected.

Click on the Update button to validate and save your schema

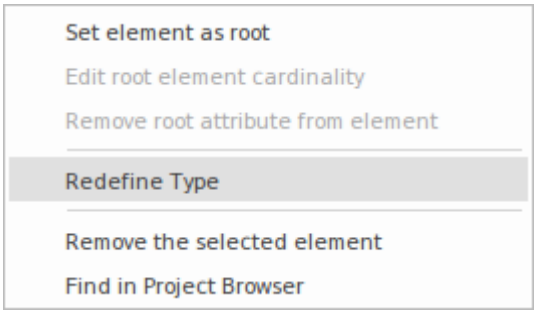
profile.

If there are any problems with the profile, they are identified in the status panel in the top right of the screen.

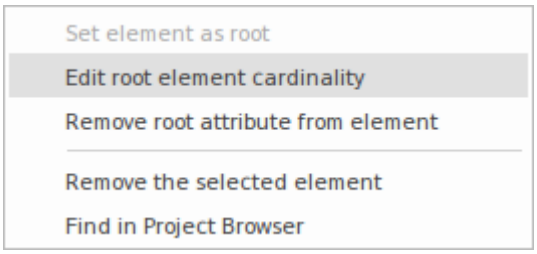
Redefined Types

One of the common issues with schema composition is the requirement to be able to vary a type description to meet various demands of the instances a schema describes. A vehicle, for example, might be described by its *brand, model and price* by an element of a *Truck* type, but by its *year, model and color* by an element of a *Sedan* type. The issue is that we might only have one actual *Vehicle* Class at our disposal. To address this the Schema Composer allows you to clone the *Vehicle* Class and give it another name. You can then assign this version of *Vehicle* to any property that has *Vehicle* as its type. The type created is only available within the domain of the schema - the model is untouched.

To create a new definition of a type, select the Class first in the 'Classes' list, then right-click on it and choose the 'Redefine Type' option. Enter a unique name for this type and press the Enter key. You can then define or restrict this type independently, the way you would for any Class.



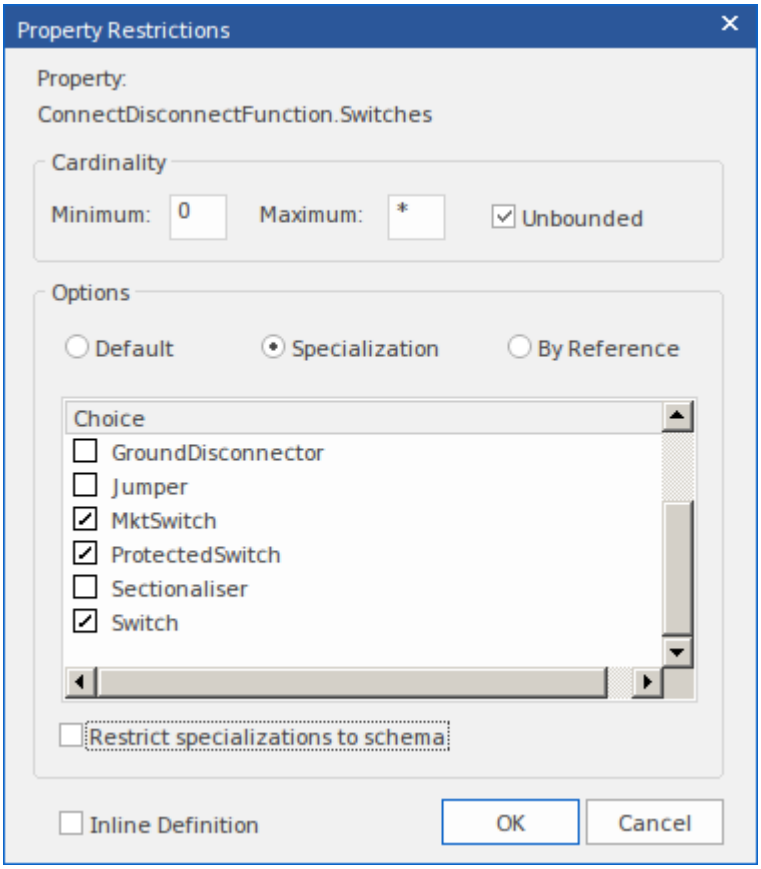
Root elements



When the schema is generated, a single top level element representing the message is generated. The body or elements of this top level element are the Classes marked as root elements. The cardinality of these root elements can be adjusted. To mark a Class as a root element or restrict its cardinality, right-click on the Class in the list and use these context menu options:

- Set element as root - root elements form the body of the top level element representing the message / profile
- Edit root element cardinality - set the minimum and maximum number of instances
- Remove root attribute from element - removes the root mark from the Class
- Remove the selected element - delete the selected element from the schema
- Find in Project Browser - locate and highlight the element in the Browser window

Property Restrictions



In the 'Attributes' list, right-click on a selected property and use the context menu to add, edit or remove a property restriction. Use this feature to:

- Modify the property cardinality
- Redefine the type of the property
- Enable and limit the choices available for this property
- Mark a property to be emitted as an inline element definition
- Mark a property to be emitted 'By Reference'

Cardinality

The cardinality of a property can be further restricted from its model counterpart, but it cannot be less restrictive. The cardinality can be changed for any root element Class and

any Class property.

Type redefinition

When a Class is redefined within the Schema Composer it creates a new type. The new type is a clone of the original, but has a name that is unique to the schema. A Payment enumeration type, for example, might be redefined as a CardPayment to better suit the schema purpose. The new type is a restriction of the original in that no new attributes can be added to it. Other properties that share this type and be similarly restricted by specifying the new type in their restriction dialog. Redefined types such as sub types can be offered as additional choice elements in the restriction of other properties.

Specializations

Where specializations of a property's type are present, those subtypes will be available in the 'Restriction' dialog. When more than one specialization is selected, these will appear as choice elements in the schema. When only one is chosen, the property will exhibit this subtype in the schema.

Inline Elements

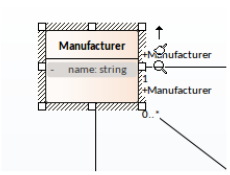
A property type will be emitted as an inline definition when this box is checked.

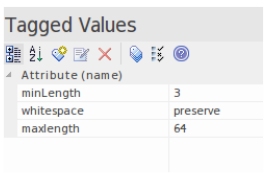
By Reference

A property will take the 'By reference' form when emitted in the schema. The 'By reference' form emits an inline complexType that defines a single attribute named 'ref' of type 'string'.

Property Constraints - Facets

Facets are supported in the Schema Composer Generic Profile. The sources of facets are the Tagged Values on a property. Tagged Values are recognized as facets if they name a constraining facet from the XML Schema specification; JSON validation keywords are also recognized.






```
<xs:complexType name="Manufacturer">
    <xs:sequence>
        <xs:element name="name" minOccurs="1" maxOccurs="1">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:maxLength value="64"/>
                    <xs:minLength value="3"/>
                    <xs:whiteSpace value="preserve"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
```

Constraining Facets from XML Schema:

- length
- minLength
- maxLength
- pattern
- enumeration
- whiteSpace
- maxInclusive
- maxExclusive

- minExclusive
- minInclusive
- totalDigits
- fractionDigits

Validation keywords in JSON:

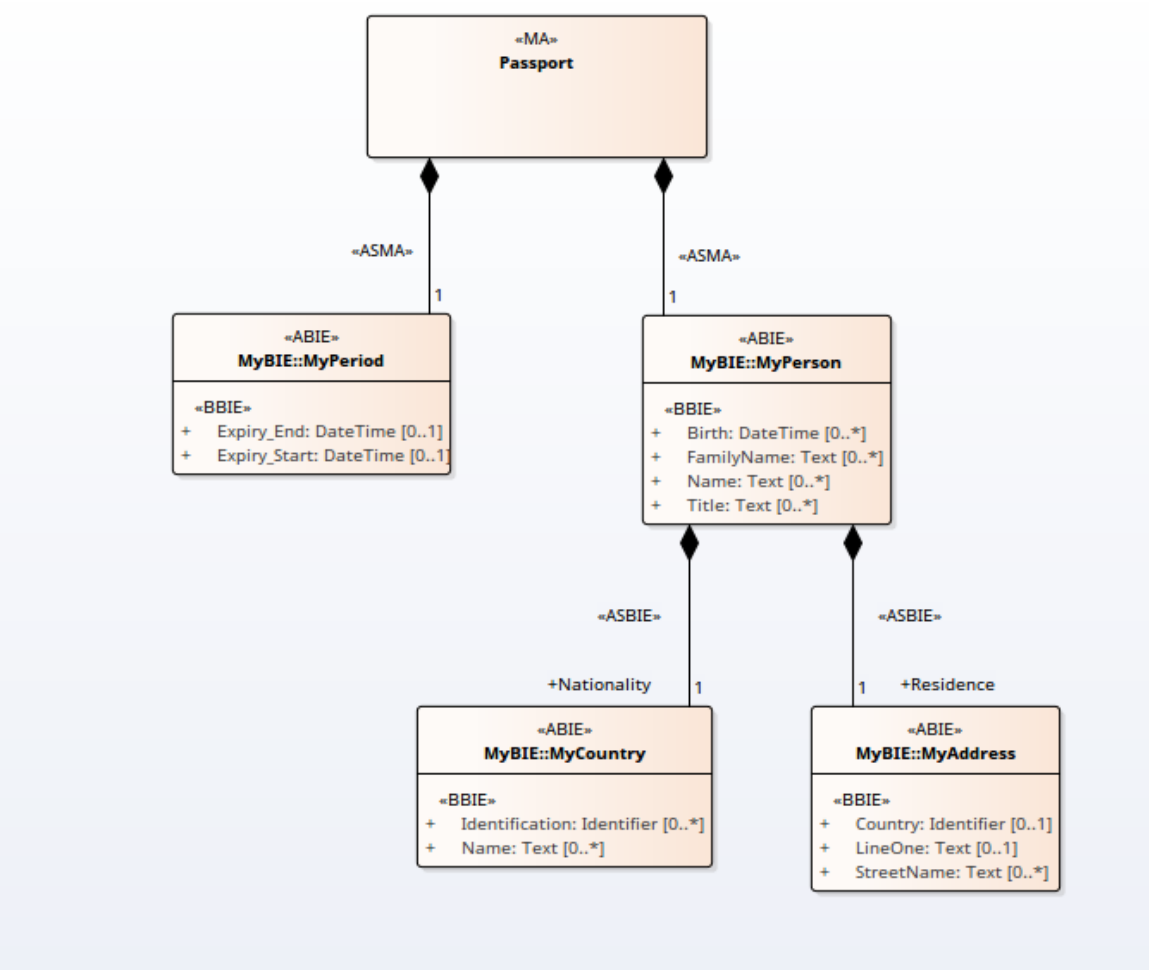
- Number and integer
  - multipleof
  - minimum
  - maximum
  - exclusiveMinimum
  - exclusiveMaximum
- strings
  - minLength
  - maxLength
  - pattern
- arrays
  - minItems
  - maxItems
  - uniqueItems

Class Diagrams

The Schema Composer also supports the creation of simple XSD and other formats from generic UML Classes. This is particularly useful when there is a need to export a Class definition in a generic manner for consumption by a script or web based tool, for example.

Generating schema from Class diagram

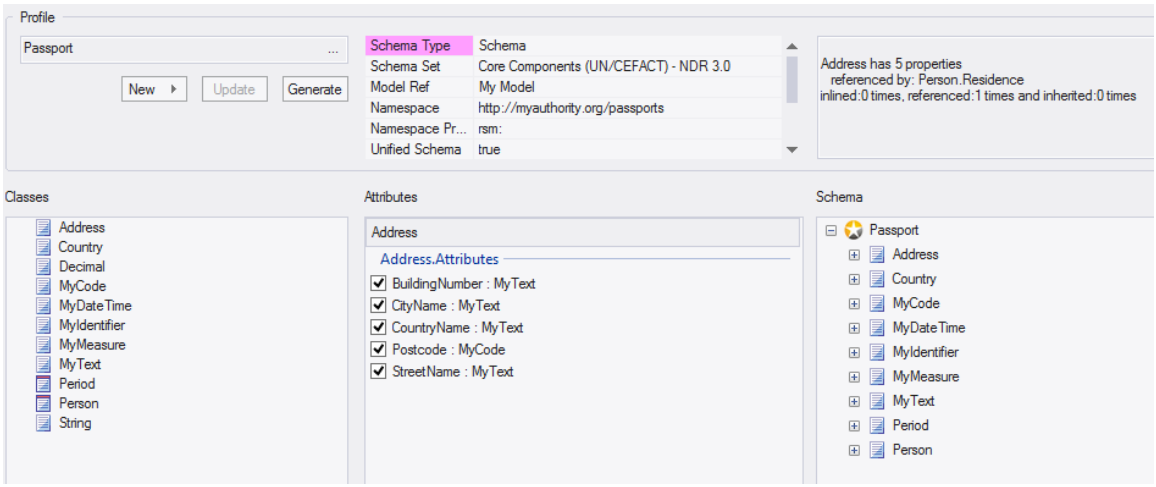
Users who prefer to use a modeling approach in composition can also use the Schema Composer for the generation of their chosen format(s). Any Class diagram can be loaded into the Schema Composer. This image illustrates a message composed using the UML Profile for Core Components, but it is not necessary for the message to be modeled according to a particular UML profile.



Loading the message into the Composer

The message is loaded into the Composer by selecting a Class on the diagram that represents the message and using its context menu to present the diagram as a schema in the Schema Composer. The selected Class will become the root element of the message and its relationships will shape the schema that is loaded.

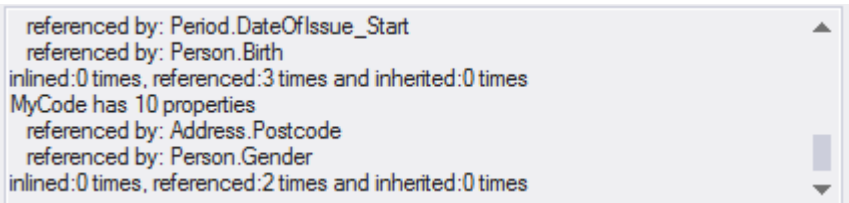
This is the Class diagram loaded into the Schema Composer


Profile
Passport
New
Update
Generate
Schema Type
Schema
Schema Set
Core Components (UN/CEFACT) - NDR 3.0
Model Ref
My Model
Namespace
http://myauthority.org/passports
Namespace Pr...
rsm:
Unified Schema
true
Address has 5 properties
referenced by: Person.Residence
inlined:0 times, referenced:1 times and inherited:0 times
Classes
Address
Country
Decimal
MyCode
MyDateTime
MyIdentifier
MyMeasure
MyText
Period
Person
String
Attributes
Address
Address.Attributes
BuildingNumber : MyText
CityName : MyText
CountryName : MyText
Postcode : MyCode
StreetName : MyText
Schema
Passport
Address
Country
MyCode
MyDateTime
MyIdentifier
MyMeasure
MyText
Period
Person

Schema Analysis

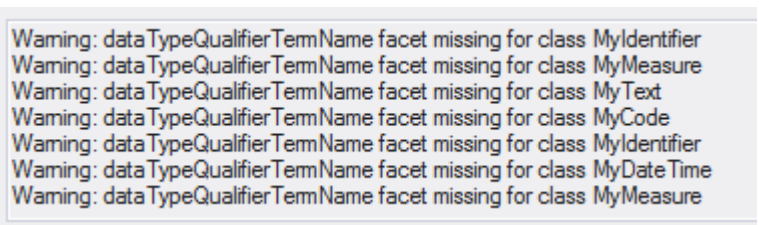
Analysis on the go

The Schema Composer performs analysis of each type as it is added to the schema, and whenever the Class is selected. The System Output window will show how many, if any, references exist for the type, the number of times it is inherited and other helpful information. This illustration shows a message detailing the elements that are referencing the selected Class.



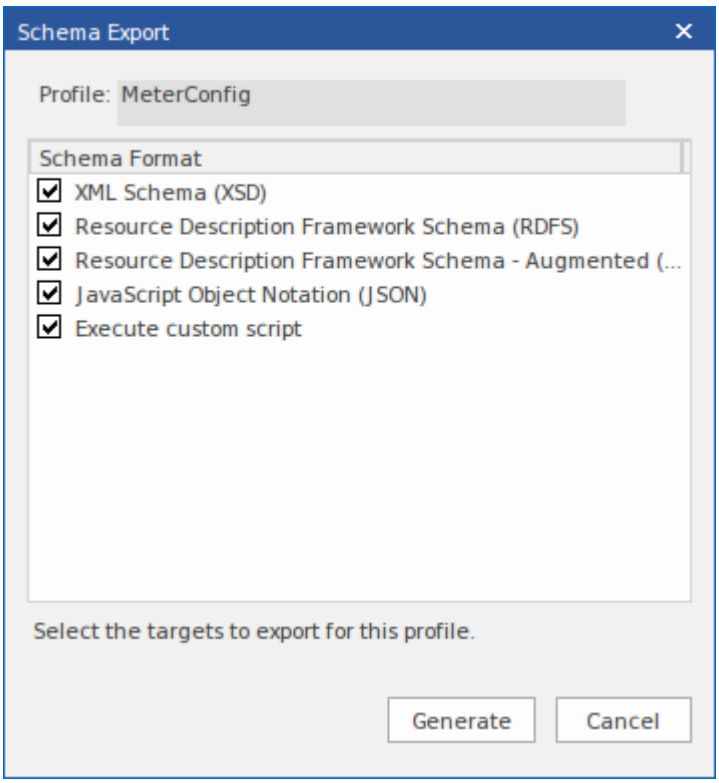
Validation on the go

The Schema Composer performs specific validation for a technology should one be assigned. This image shows warnings about missing Tagged Values for Classes in a schema built on the UN/CEFACT Core Components standard.



Generate Schema

Having designed a profile, at any stage, with a minimum of definitions and customizations, you can quickly and easily generate the schema or sub-model. Depending on the technology chosen and the profile type (schema or transform), the formats presented to you will vary. Note multiple formats can often be generated at once. And, of course, you can repeat the process easily, as the composition evolves or after design changes to the model.



Access

Ribbon	Develop > Schema Modeling > Schema Composer > Open Schema Composer :

	Generate

Notes

- The Schema Composer is supported in the Corporate, Unified and Ultimate Editions of Enterprise Architect

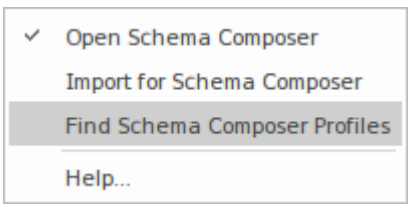
Select a Schema Profile

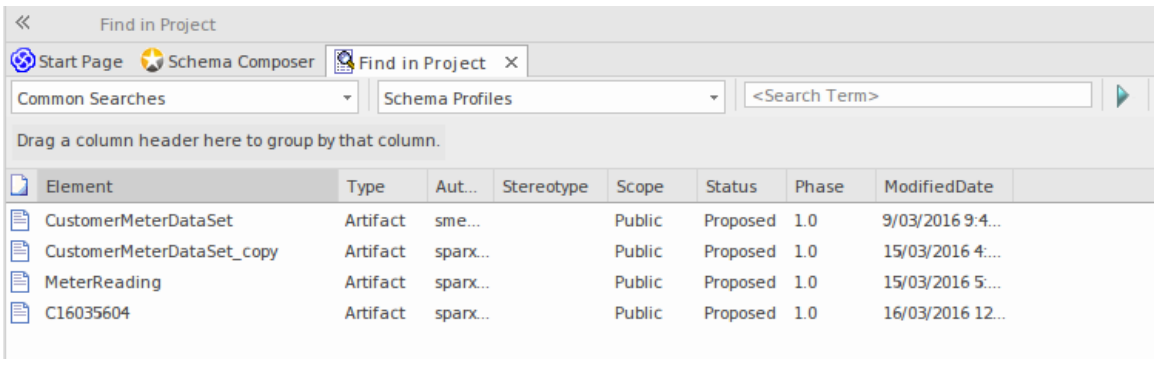
Access

Ribbon	Develop > Schema Modeling > Schema Composer > Find Schema Composer Profiles Develop > Schema Modeling > Schema Composer (icon)

Locating Schema Profiles in the Model

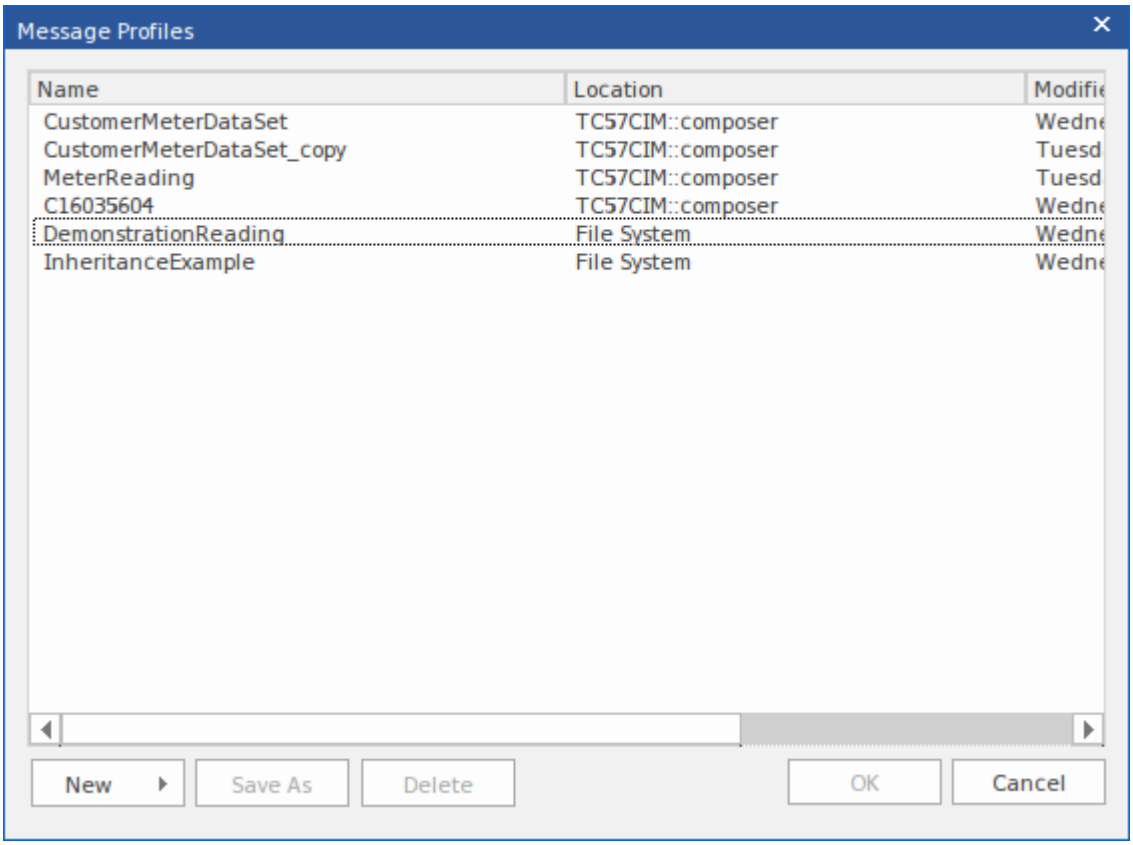
Schema profiles can be located quickly from the Schema Composer drop-down menu in the ribbon. The menu provides quick access to the existing profiles in the model.





Locating Schema Profiles in the Schema Composer

Schema profiles can stored in either the model and file system. You can easily locate every profile by authored in your model by opening the Schema Composer and clicking the select profile button (the button with the ellipsis '...') This brings up a list of all the profiles for this model and indicates where they live; the model or the file system. You might have many Enterprise Architect models that you work with, but only those file system profiles that relate to the open model will be listed.



Generate Schema File

Having defined a schema profile and added the necessary elements and restrictions, you can quickly and easily generate the schema(s). XML schema generation is available in all technologies, but each technology might support additional formats.

Access

Ribbon	Develop > Schema Modeling > Schema Composer > Open Schema Composer : Generate

Schema Formats

Select the checkbox against each schema format to export.

Schema Format	Details
CIM	• XML Schema (XSD) • Resource Description Framework Schema (RDFS) • Resource Description Framework Schema - Augmented (RDFS)

	• JavaScript Object Notation (JSON) • Execute Custom Script
UN/CEFACT NDR 3.0	• XML Schema (XSD) • Execute Custom Script
UN/CEFACT NDR 2.1	• XML Schema (XSD) • Execute Custom Script
Generic	• XML Schema (XSD) • Resource Description Framework Schema (RDFS) • JavaScript Object Notation (JSON) • Execute Custom Script
UBL 2.1	• XML Schema (XSD) • Execute Custom Script
Execute Custom Script	Although the Schema Composer can generate schema for a number of recognized standards, it also features a scripting solution for those users who want control over the format and medium of the schema. When you specify a script to the generator, it is referring to a language script such as JavaScript that exists in your model. How and what the script produces is pretty much up to you.

	How the script accesses the schema in the Schema Composer is documented in the Schema Composer Scripting Integration.

Generate

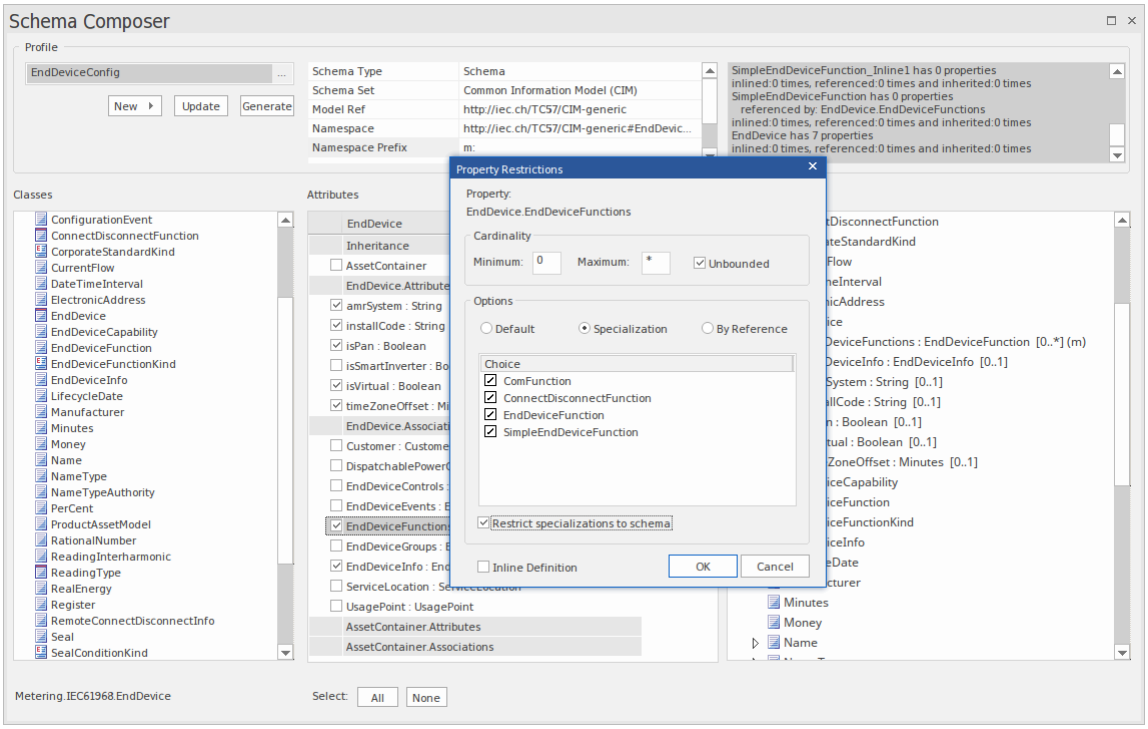
Click on this button to generate the schema.

Use a file browser to locate and open the schema files.

Notes

- You can edit and validate XML documents including XSD schema, using Enterprise Architect
- You can set Enterprise Architect as the default document handler for XML documents

CIM Schema Guide



This guide describes the creation and generation of a CIM compliant XML Schema.

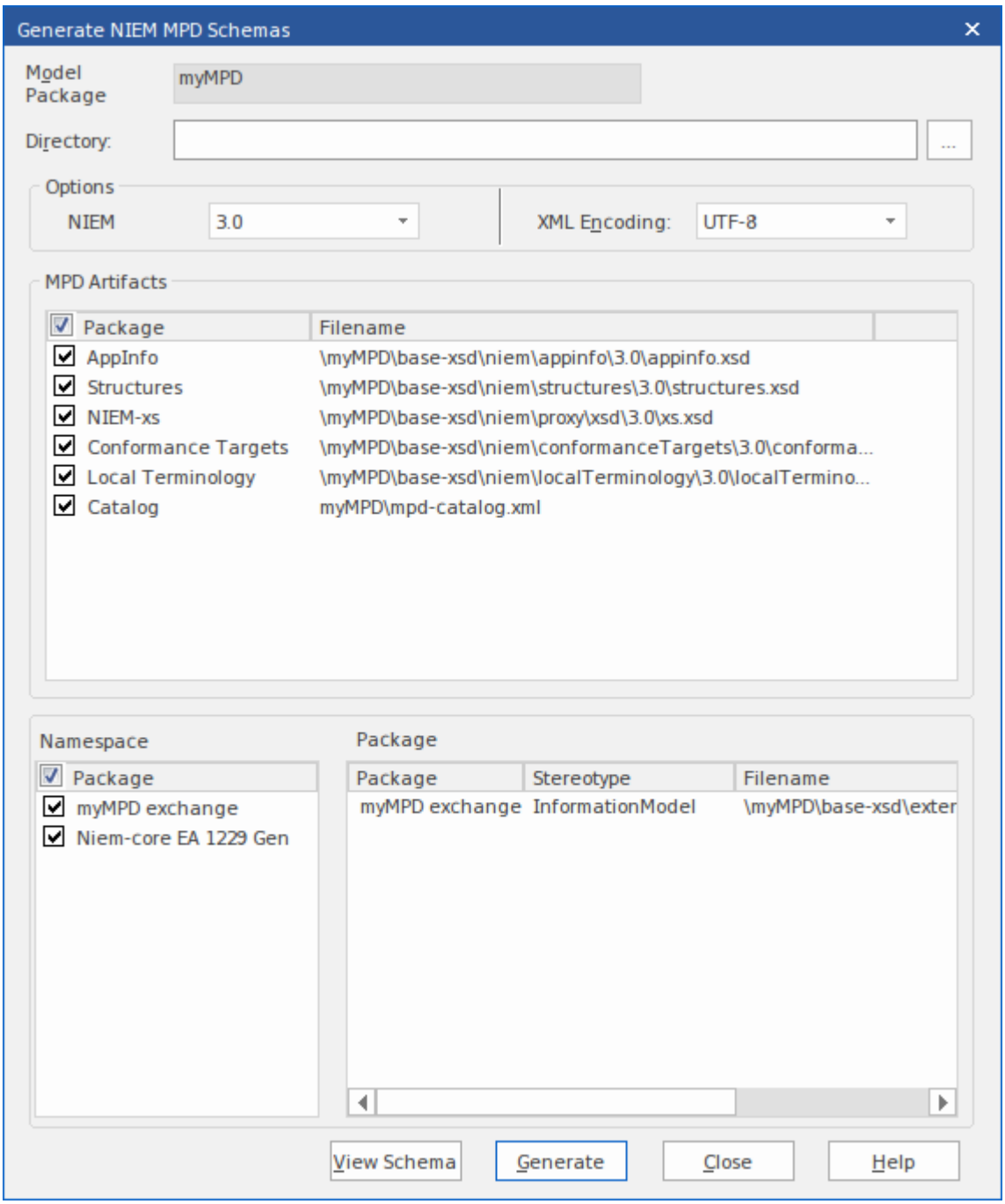
Create a CIM message

Step	Action
1	Display the Schema Composer.
2	Click 'New | Schema'.
3	Enter a unique name for this CIM schema (message).

4	Select the Common Information Model.
5	Drag the initial CIM Class(es) into the Class window that best represents the message. Set *root* elements appropriately using the context menu.
6	If you want to compose this type using inheritance, select a single ancestor from the inheritance list.
7	Use the checkboxes on the attributes of each Class to define the set of properties that will describe this message or schema.
8	Apply restrictions to elements using the context menu on the property.
9	Click update to save the message.
10	Click the Generate button and choose the schema formats to export.

NIEM Schema Guide

Generation of schema for NIEM is accomplished on either an instance of a ModelPackageDescription Class (NIEM 3.0 and above) or a «ModelPackageDescription» stereotyped component (NIEM 2.1). In either case a dialog is presented that allows you to configure the schema produced.



Generate NIEM Schemas (NIEM 2.1)

Click on a component with a «ModelPackageDescription» stereotype and select one of these options:

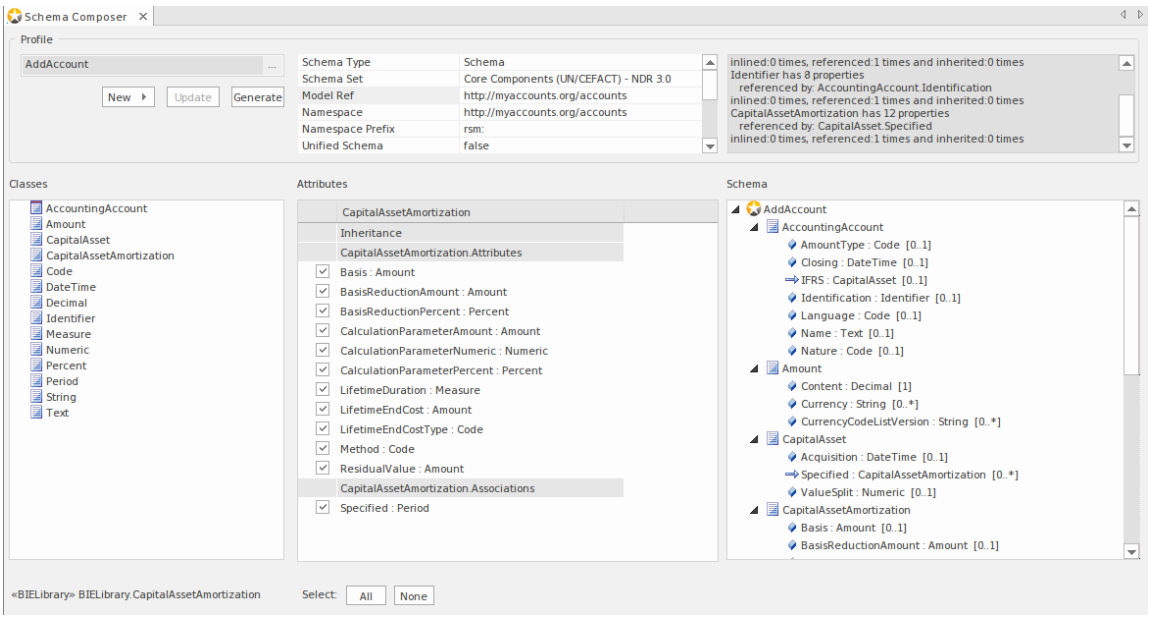
Ribbon	Specialize > Technologies > NIEM 2.1 > Generate NIEM 2.1 Schema or
Context Menu	Right-click on the element | Specialize | NIEM 2.1 | Generate NIEM 2.1 Schema

Generate NIEM Schemas (NIEM 3.0 and above)

Click on any object instance of a ModelPackageDescription Class and select one of these options:

Ribbon	Specialize > Technologies > NIEM > Generate NIEM Schema
Context Menu	Right-click on the element | Specialize | NIEM | Generate NIEM Schema

UPCC Schema Guide



This guide describes the composition and generation of a UPCC compliant XML schema.

Creation of UPCC Schema

Step	Action
1	Display the Schema Composer.
2	Click 'New | Schema'.
3	Enter a unique name for the schema.
4	Select the UPCC Naming and Design rules to use,

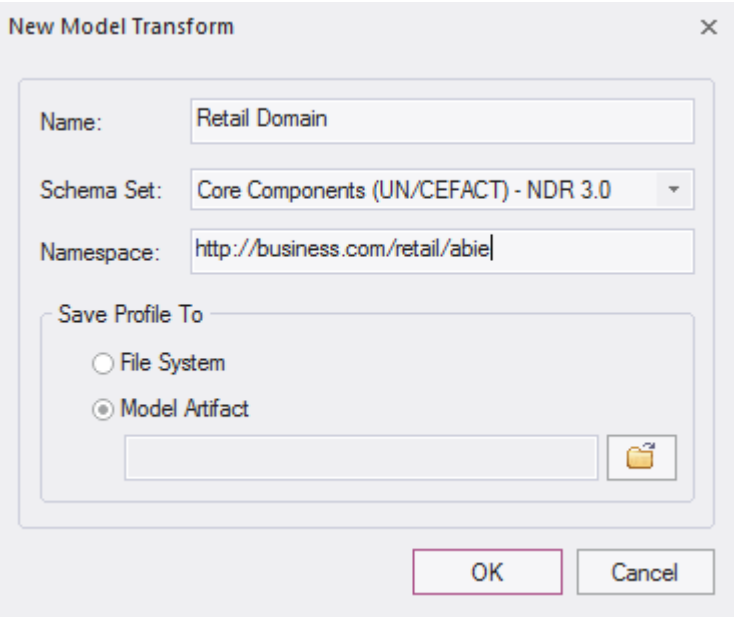
	from the list of standards.
5	Drag one or more <ABIE> components from a <BIE library> into the Class list.
6	Set the Class as a root element using the context menu.
7	Select required attributes (Referenced types are added to the schema).
8	Click on the Update button to save changes.
9	Click on the Generate button and select 'XML Schema'. Click on the OK button.

Model Compositions

The model composition feature of the Schema Composer is useful for creating a sub-model from a core model. This can be as simple as generating a single business Package from a core Package (*CDT library to BDT library transform in UN/CEFACT Core Components standard)* or creating a complete sub-model from a large core model.

The enormity of such a task can be daunting and error prone; *ensuring every type that is referenced by the sub-model is included by the sub-model,* for example. The Schema Composer addresses this problem by automatically working out the dependencies and adding them to the schema for you where necessary.

Create Transform

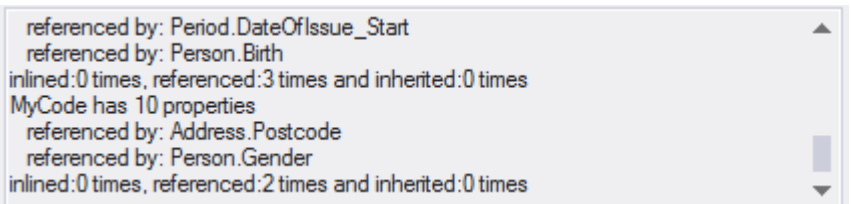


Define Model Content

Content is added to the model by dropping Classes from the model on to the Schema Composer Class window and choosing which properties to include. The resultant type can mirror the core type or a provide a simpler classification. When a property is included, the Schema Composer will check the property type and if the type is missing will add it to the schema automatically.

Reference checking

When a property is excluded that was previously included in the schema and is no longer referenced, the property type is not automatically removed. However, the Schema Composer will always show the number of references for a type if you select it in the Class window. Types that show no references at all can easily be removed.



Summary

The process of composing a sub-model is summarized here:

1. Create a Schemer Composer Transform Profile
2. Create elements by dropping Classes from the model into the schema.
3. Include / exclude required properties.

4. Generate the sub model.

Access

Ribbon	Develop > Schema Modeling > Schema Composer > Open Schema Composer : New > Transform

Generate a Model Subset (Transform)

Having defined the content of your sub-model or library and applied any restrictions, you can now generate the model. The model transforms that can be performed depend on the technology associated with the profile. Each technology and the transforms it supports are listed here:

Access

Ribbon	Develop > Schema Modeling > Schema Composer > Open Schema Composer : Generate

Model Transform

Select the model transform(s) to run.

Transform Option	Description
NIEM	**NIEM Model Subset** This option will generate a NIEM Model Subset containing the schema described by the profile. When you click on the OK button, you

	will be prompted to select the target Package for creation of the subset. The <<Namespace>> Packages that make up the subset will then be created at this location. If any of the subset Packages already exist at this location, their content will be added to. All subset Packages will have the 'defaultPurpose' Tagged Value set to 'subset'. **Execute custom script** A user defined language script such as JavaScript will be executed. The script can obtain access to the profile using the Schema Composer automation interfaces.
Generic	**Generic model Subset** You will be prompted for a target Package. This will be populated with the types from the schema. If a qualifier is entered this will be applied to the Classes generated. Any restrictions in the schema will also be applied. Types that exist in the target Package will be overwritten. New properties will be added. Types or properties that exist in the target but that no longer exist in the profile will not be removed by this process. **Execute custom script**

	A user defined language script such as JavaScript will be executed. The script can obtain access to the profile using the Schema Composer automation interfaces.
UN/CEFACT NDR 3.0	**BDT Library** A Business Datatype Library will be populated from core datatypes listed in the profile. Stereotypes will be transformed according to the CCTS specification. The types could be more restricted than their core counterparts. Properties of datatypes that exist will be overwritten. New properties and types will be added to the library. Types are matched by name and stereotype. Types or properties that exist in the target but no longer exist in the profile will not be removed by this process. **BIE Library** A Business Information Entity library will be populated from aggregated core components. listed in the profile. Stereotypes will be transformed according to the CCTS specification. Properties of datatypes that exist will be overwritten. New properties and types will be added to the library. Types are matched by name and stereotype.

	Types or properties that exist in the target but no longer exist in the profile will not be removed by this process. **Execute custom script** A user-defined language script such as JavaScript will be executed. The script can access the profile using the Schema Composer automation interfaces.
UN/CEFACT NDR 2.1	**UDT Library** Performs an unqualified copy of selected core datatypes to a UDT library. **QDT Library** A Qualified Business Datatype Library will be populated from core datatypes listed in the profile. The names of the resultant types will be qualified by the named qualifier in the profile. Stereotypes will be transformed according to the CCTS specification. Properties of datatypes that exist will be overwritten. New properties and types will be added to the library. Types are matched by name and stereotype. **BIE Library** A Business Information Entity library will be populated from aggregated core components listed in the profile.

	Stereotypes will be transformed according to the CCTS specification. Properties of datatypes that exist will be overwritten. New properties and types will be added to the library. Types are matched by name and stereotype. **Execute custom script** A user-defined language script such as JavaScript will be executed. The script can obtain access to the profile using the Schema Composer automation interfaces.

Generate

Click on the OK button to generate the schema. When the generation is complete, the message *Export of profile <name> completed* displays.

You can then expand the Package in the Browser window to see the generated UML model.

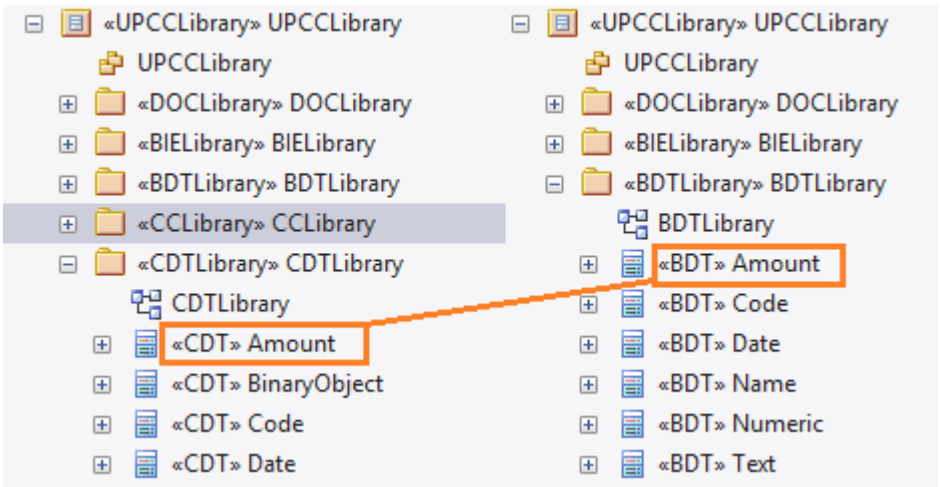
Notes

- The Schema Composer is supported in the Corporate, Unified and Ultimate Editions of Enterprise Architect

UML Profile for Core Components (UPCC)

The UPCC framework provides core component and core data type libraries and is available to Enterprise Architect users through the Model Builder. Whether you model according to the UMM specification, or want to leverage the advantages this standard brings, or have a compliance requirement, to model with this technology you will require - as a minimum - a Business datatype library and a Business Information Entity library. The Schema Composer can generate these libraries for you.

This image shows a BDT library created from a UPCC Core CDT Library



Common Libraries

Libraries shared by both versions of the UML Profile for Core Components.

Library	Description

CCLibrary	The CCTS core component library.
CDTLibrary	The CCTS core datatype library. It contains basic datatypes such as Amount, Code, Text and Graphic.
BIELibrary	A Business library containing ABIE entities based on CCLibrary components. The entities can be composed using the Schema Composer. These can also be modeled using the UML modeling tools available for the technology.
DOCLibrary	A Package typically used for the modeling of Message Assemblies. You can generate the schema for a Message Assembly by loading it into the Schema Composer.-

UPCC Libraries

The UML Profile for Core Components is available in two versions, NDR 3.0 and NDR 2.1. Both profiles describe a common set of libraries, with some differences, as described here:

NDR 3.0

Library	Description
BDTLibrary	A Business library containing BDT types based on CDTLibrary types. The Schema Composer can be used to easily generate the content of a BDTLibrary from selected types in the core library.

NDR 2.1

Library	Description
UDTLibrary	An unqualified datatype library. Basically a mirror of the CDTLibrary for use in a business context. The Schema Composer can be used to easily generate the content of a UDTLibrary from selected types in the core library.
QDTLibrary	A qualified datatype library. The library contains restricted types based on the CDTLibrary with qualified type names. The Schema Composer can easily generate the content of a QDTLibrary

	from selected types in the core library.

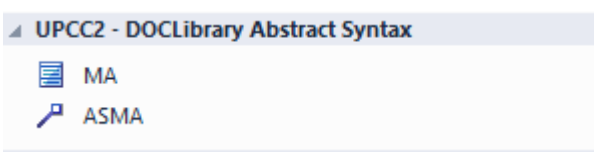
UPCC Diagrams

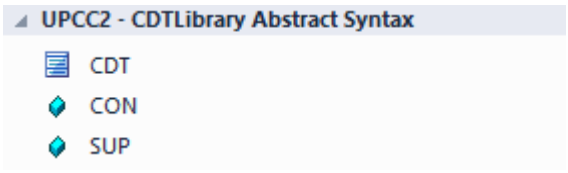
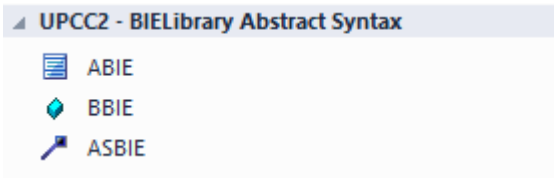
The UML profile for Core Components uses UML Class diagrams for composition of components. There are however specific Toolbox pages provided by the technology for each of its libraries.

UPCC Toolbox Pages

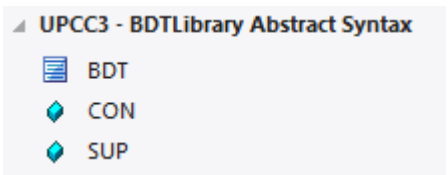
Common

In this notation, UPPCx represents the UPCC profile and x is the version of the NDR

Common	Description
UPCCx - CCLibrary Abstract Syntax	
UPCCx - DOCLibrary Abstract	

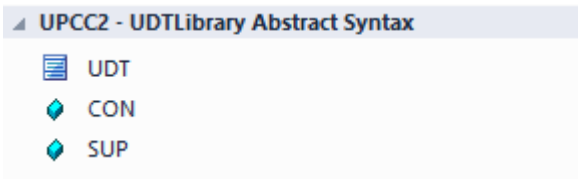
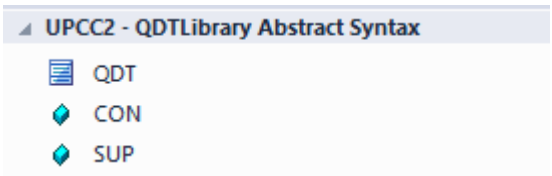
Syntax	
UPCCx - CDTLibrary Abstract Syntax	
UPCCx - BIELibrary Abstract Syntax	

NDR 3.0

Library Syntax	Description
UPCC - BDTLibrary Abstract Syntax	

NDR 2.1

Library	

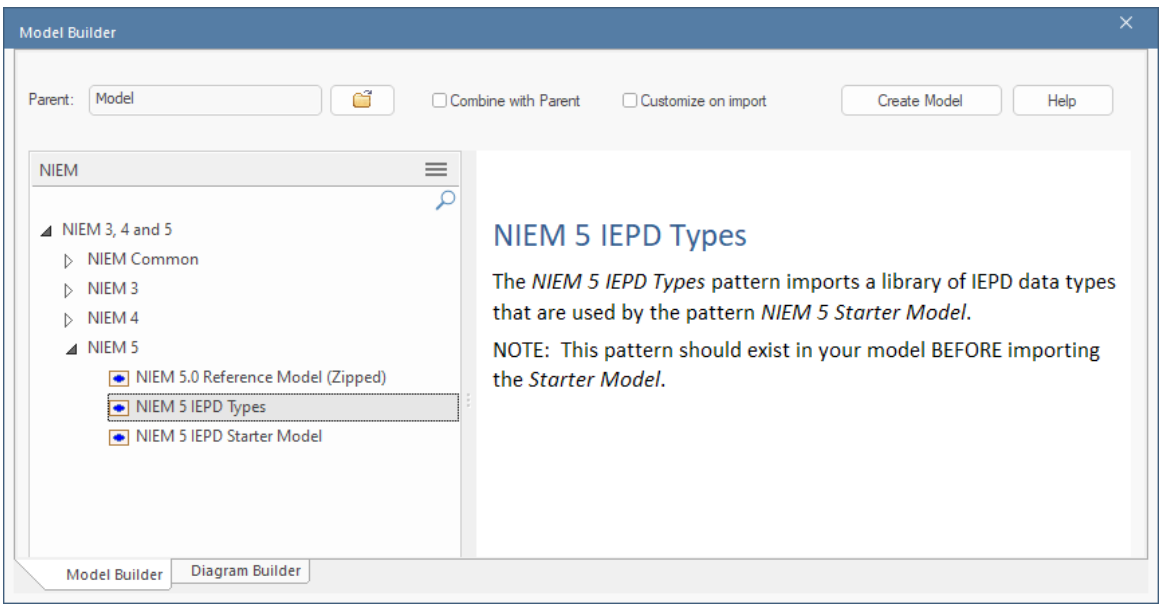
Syntax	Description
UPCC - UDTLibrary Abstract Syntax	
UPCC - QDTLibrary Abstract Syntax	

Available Frameworks

Using the Enterprise Architect Model Builder you can deploy any of the frameworks supported by the Schema Composer - such as NIEM, CIM and CCTS - to your model in minutes, providing a multi-featured UML medium for modeling in those technologies.

The frameworks are also available directly from the Sparx Systems Reusable Asset Service (via the Cloud Server 'Cloud Connection' dialog and then 'Frameworks' on the Model Builder).

Note: In addition to the custom frameworks such as CIM and NIEM, it is possible to use standard Class models to rapidly build generic Schemas, so if you are not targeting a particular meta-model, it might be simplest to model your data in UML and use the resultant model as input to the Schema Composer.



National Information Exchange Model

(NIEM)

This is the National Information Exchange Model published by the NIEM Program Management Office (PMO).

Enterprise Architect provides these resources for modeling in NIEM:

- Provided Frameworks including NIEM core, NIEM domains, code lists and external schema adapters:
  - NIEM 3.0 modeled using NIEM-UML 1.1
  - NIEM 3.1 modeled using NIEM-UML 1.1
  - NIEM 4.0 modeled using NIEM-UML 1.1
  - NIEM 5.0 modeled using NIEM-UML 1.1
- NIEM subset creation:
  - The Schema Composer helps you create a subset of a NIEM conformant namespace
- NIEM schema generation:
  - Generation of complete NIEM IEPDs from a model Package description in either NIEM 3, NIEM 4 or NIEM 5 formats

Common Information Model (CIM)

This is the CIM specification published by International Electrotechnical Commission (IEC) Technical Committee 57.

Enterprise Architect provides these resources for modeling in CIM:

- Schema Composition

- XML schema (XSD)
- Resource Descriptor format (RDFS)
- Resource Descriptor augmented format
- JavaScript Object notation (JSON)
- Add-In integration
- Scripting integration

Universal Business Language (UBL)

UBL is a CCTS implementation published by OASIS that is proving popular with European governments for consolidating information exchange between agencies.

Enterprise Architect provides these resources for the composition of business documents using UBL:

- UML Framework
  - UBL 2.1 Main Document Libraries
  - UBL 2.1 Common Component Libraries
- Business Document Composition
  - Schema Composer for component composition
  - Schema Composer for document composition
  - Schema Composer for schema generation
  - Add-In integration
  - Scripting integration

Core Component Technical Specification (CCTS)

This is the CCTS specification published by UN/CEFACT.

Enterprise Architect provides these resources for modeling in CCTS:

- UML Frameworks:
  - UPCC 2.1 core component libraries
  - UPCC 3.0 core component libraries
  - UMM 2.0 business requirements, choreography and information views.
- Business Component Library Creation / Management
  - Schema Composer for ABIE and BDT composition
  - Add-In integration
  - Scripting integration
- Business Component Schema Composition
  - Schema Composer for XSD, JSON
  - Add-In integration
  - Scripting integration

Add-In Framework (Custom)

In addition to these methodologies, the Schema Composer integrates with the Enterprise Architect Automation Interface to support any individual or group in implementing their own. An Add-In that registers its interest to Enterprise Architect in offering Schema generation capabilities will have the opportunity to offer any of its products in the Schema Composer Generation tool.

Scripting Framework (Custom)

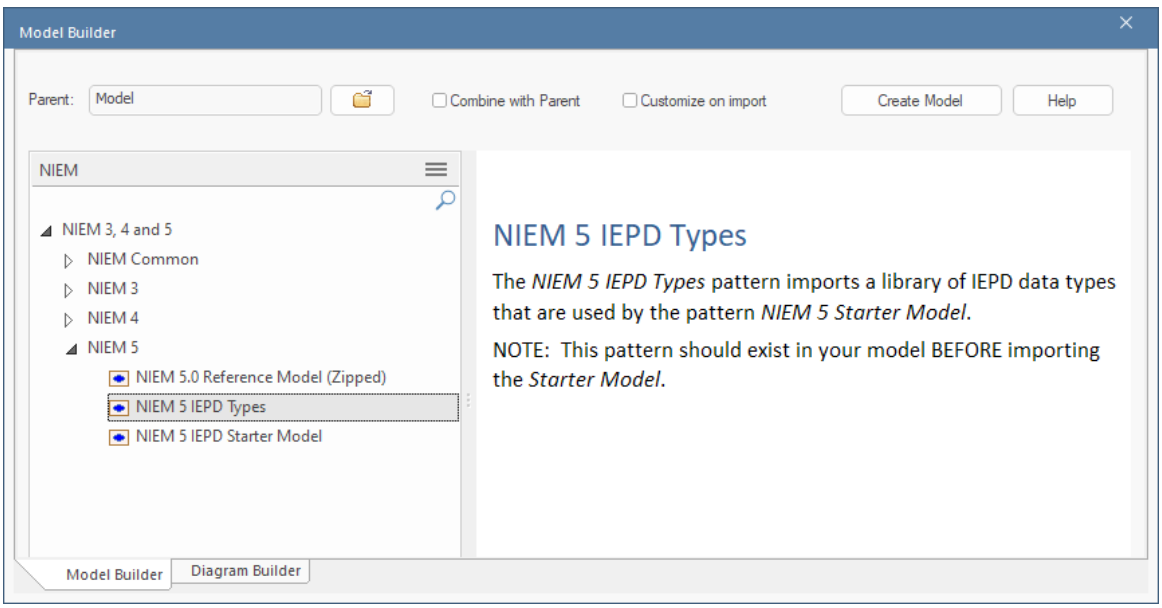
The Schema Composer also offers unconditional control over generation of schema for any profiles created with it. By writing their own script an author can access the definition of any schema and ultimately produce whatever documents they want, in a format of their choosing.

Install a Core Framework

Enterprise Architect provides a rich and diverse range of modeling technologies including every standard listed in the Schema Composer. These frameworks are available as UML models and/or MDG Technologies using Enterprise Architect's Model Builder. The models themselves are also directly accessible from Enterprise Architect's Reusable Asset Service.

Note: If you are modeling a generic solution and not directly using a core framework such as CIM or UBL, you do not need to install a core framework/model. In that case you are best served creating a data model using simple UML Classes with attributes.

Model Builder



Access

Ribbon	Start > Personal > Model Builder Design > Package > Model Builder
Context Menu	Right-click on Package | Model Builder (pattern library)
Keyboard Shortcuts	Ctrl+Shift+M
Other	Browser window 'Hamburger' icon > Model Builder (pattern library)

Note

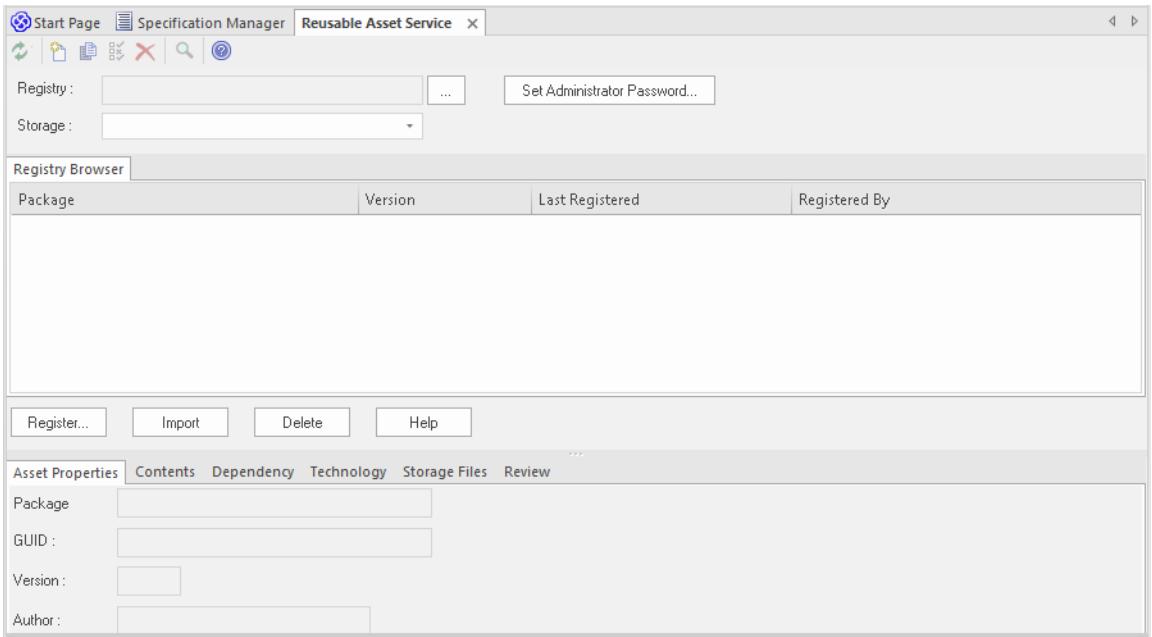
You can limit the MDG Technologies to use by selecting the ribbon option: 'Specialize > Technologies > Manage Technology'. Here you can see which technologies are currently enabled.

Import Model

Step	Action

1	Display the Model Builder dialog.
2	Click on the <perspective name> button and select the required technology.
3	Highlight the Technology.
4	Select the Technology standards to import.
5	Click on the Create Model button to import the framework to your model.

Reusable Asset Service



Access

Ribbon	Publish > Model Exchange > Reusable Assets

Import Model

Step	Action
1	Connect to the Reusable Asset Service.
2	Choose from the available list of Repositories
3	Select the UML model Package
4	Click OK to import the selected Package to your model.

The Schema Importer

You can import Schemas compatible with the Schema Composer, into Enterprise Architect using the Schema Importer. The Schema Importer validates the Schema and creates a *Schema* type Schema Composer Profile upon successful validation, that can be viewed directly in the Schema Composer.

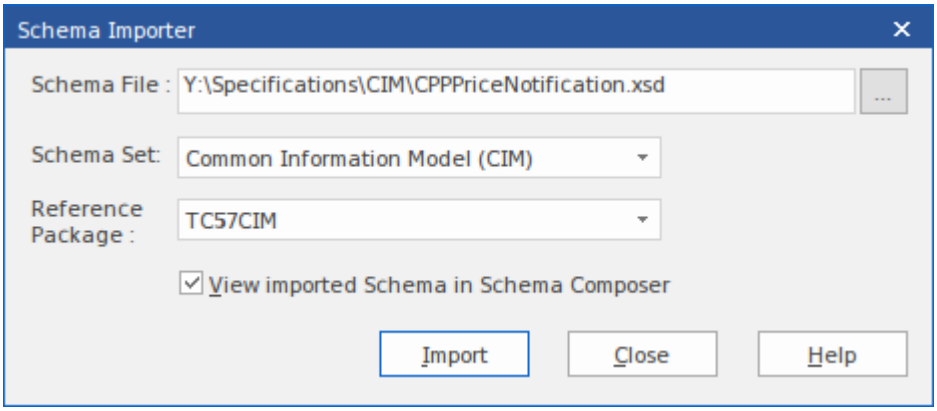
Currently, you can use the Schema Importer to import these Schemas:

- Common Information Model (CIM) specific XML Schema
- Common Information Model (CIM) specific RDFS XML

Access

Ribbon	Develop > Schema Modeling > Schema Composer > Import for Schema Composer

Import a Schema using the Schema Importer



Option	Action
Schema File	Type the directory path and filename from which to import the Schema file.
Schema Set	Select the type of Schema being imported. Currently the Schema Importer supports importing CIM specific: • XML Schema and • RDFS XML
Reference Package	Select the Package containing the common elements specific to the schema set. The Schema Importer will validate the elements in the Schema being imported against the elements in the reference Package.
View imported	Select this option to open the imported Profile in the Schema Composer.

Schema in Schema Composer	
Import	Click on this button to start the import process.
Close	Click on this button to close the 'Schema Importer' dialog.

Notes

- The progress of import will be displayed in the System Output window
- The Schema Composer will validate the Schema against the elements in the Reference Package before importing the Schema; if validation fails, the Schema elements that fail validation will be displayed in the System Output window and the import process will stop
- Double-click on a validation error entry in the System Output Window to open the Schema in Enterprise Architect's internal file editor and go to the source of the error
- If validation succeeds, the 'New Schema Definition' dialog displays, through which you can save the imported Profile in the file system or as an Artifact in the current model

Schema Composer Automation Integration

The Schema Composer can be accessed from the Enterprise Architect Automation Interface. A client (script or Add-In) can obtain access to the interface using the 'SchemaComposer' property of the 'Repository' object. This interface is available when a Schema Composer has a profile loaded.

Schema Composer Addin Integration

Enterprise Architect Add-Ins can integrate with the Schema Composer by providing alternatives to offer users for the generation of schemas and sub models.

Schema Composer Scripting Integration

Although the Schema Composer provides out-of-the-box schema composition based on a variety of popular technologies, its scripting integration provides you with some flexibility in how you might go about implementing your own requirements. There are three ways in which you might leverage scripting within the Schema Composer:

- Provide custom schema generation using a scripting language
- Provide custom model transformation using a scripting language
- Provide custom stereotype mapping to any standard model transform (such as UPCC)

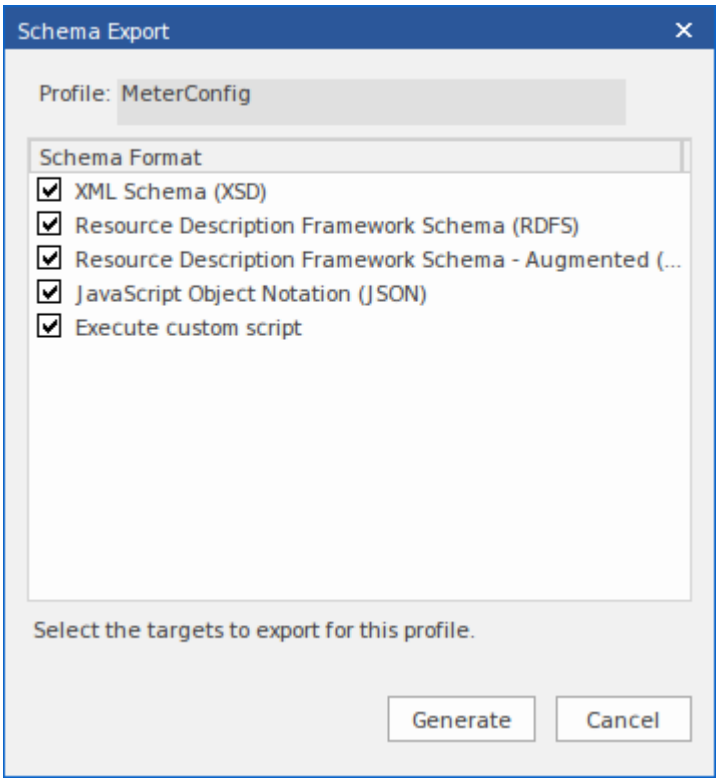
Model Transformation by script

While the Schema Composer provides in-built transforms for various frameworks, you can always write your own, using the composition tools of the Composer to design the schema, then performing a custom transform with a hand crafted script.

Schema Generation by script

When you select a message in the Schema Composer and click generate, you are presented with a number of export

formats. One of those choices is 'Execute custom script'



Schema Iteration Scripting Example

This example demonstrates accessing the Schema Composer in an Enterprise Architect script written in JavaScript. The script first obtains an interface to the Schema Composer and then traverses the schema, printing out the types and each of its properties.

```
/*
* Script Name: Example Schema Composer Script
* Author:  Sparx Systems
* Purpose: Demonstrate access to Schema Composer using
```

```
automation and JavaScript
* Language: JavaScript
* Date:  2020
*/
function printType( xmlType, xmlns, uri)
{
	var xmlProp as EA.SchemaProperty;
	var xmlPropEnum as EA.SchemaPropEnum;
	var xmlChoiceEnum1 as EA.SchemaTypeEnum;
	var xmlChoiceEnum2 as EA.SchemaTypeEnum;

	Session.Output("Type: " + xmlType.TypeName + " in
namespace: "  + xmlns + ":" + uri);
	xmlPropEnum = xmlType.Properties;
	if(xmlPropEnum)
	{
		xmlProp = xmlPropEnum.GetFirst();
		while(xmlProp)
		{
			if(xmlType.IsEnumeration())
			{
				Session.Output("   " + xmlProp.Name);
			}
			else
			{
```

```
                var sPropDesc = xmlProp.Name;
                sPropDesc += "::"
                if(xmlProp.IsPrimitive())
                    sPropDesc +=
xmlProp.PrimitiveType;
                else
                    sPropDesc += xmlProp.TypeName;

                if(xmlProp.IsByReference())
                {
                    sPropDesc += "(by reference)";
                }
                if(xmlProp.IsInline())
                {
                    sPropDesc += "(inline)";
                }
                Session.Output("    " + sPropDesc + ",
cardinality: " + xmlProp.Cardinality);

                xmlChoiceEnum1 = xmlProp.Choices;
                xmlChoiceEnum2 =
xmlProp.SchemaChoices;
                var count = xmlChoiceEnum1.GetCount()
+ xmlChoiceEnum2.GetCount();
                if(count>1)
```

```
                    {
                        Session.Output("   choice of: ");
                        xmlChoice =
xmlChoiceEnum1.GetFirst();
                        while(xmlChoice)
                        {
                            Session.Output("     " +
xmlChoice.TypeName);
                            xmlChoice =
xmlChoiceEnum1.GetNext();
                        }
                        xmlChoice =
xmlChoiceEnum2.GetFirst();
                        while(xmlChoice)
                        {
                            Session.Output("     " +
xmlChoice.TypeName);
                            xmlChoice =
xmlChoiceEnum2.GetNext();
                        }
                    }
                }
                xmlProp = xmlPropEnum.GetNext();
            }
        }
}
```

```
function main()
{
	var schema as EA.SchemaComposer;
	var xmlType as EA.SchemaType;
	var xmlTypeEnum as EA.SchemaTypeEnum;
	var xmlNamespaceEnum as
EA.SchemaNamespaceEnum;
	var xmlNS as EA.SchemaNamespace;

	// Get SchemaComposer
	schema = Repository.SchemaComposer;

	// print the namespace references
	xmlNamespaceEnum = schema.Namespaces;
	if(xmlNamespaceEnum)
	{
		xmlNS = xmlNamespaceEnum.GetFirst();
		while(xmlNS)
		{
			Session.Output( "xmlns:" + xmlNS.Name + "
URI=" + xmlNS.URI);
			xmlNS = xmlNamespaceEnum.GetNext();
		}
	}
```

```
    // Get Schema Types Enumerator
    xmlTypeEnum = schema.SchemaTypes;
    xmlType = xmlTypeEnum.GetFirst();
    while(xmlType)
    {
        var xmlns = schema.GetNamespacePrefixForType(
xmlType.TypeID );
        uri = schema.GetNamespaceForPrefix(xmlns);
        printType(xmlType, xmlns, uri);
        xmlType = xmlTypeEnum.GetNext();
    }
}

main();
```

Intelli-sense help in scripting

The Scripting editor in Enterprise Architect will help you write script that interacts with the Schema Composer, by providing Intelli-sense on the properties and methods of its Automation Interface.

```
// Enumerate Types
xmlType = umlModelTypeEnum.GetFirst();
while(xmlType)
{
    print( "Type: " + xmlType.TypeName );

    umlPropEnum = xmlType.
    if(umlPropEnum)
    {
        umlProp = umlPropE
        while(umlProp)
        {
            if(umlProp.IsP
```

GetFacet
GetRestriction
GUID
Properties
PropertyCount
TypeID
TypeName

Stereotype mapping in Model Transformation

Stereotyping forms a large part of the MDG Technology approach. Individual UML profiles for an MDG Technology define stereotypes to offer useful classifications for its elements. It is a common requirement when going from a core framework to a business model or sub-domain to reassign the stereotype. When you work with a CCTS framework the business components you generate have their stereotype automatically generated by Enterprise Architect according to a mapping defined by the CCTS specification (ACC to ABIE, for example).

When you open or create a model transform profile in the Schema Composer you can specify a script to perform this mapping for you. The script can be selected from the Properties window.

Schema Type	Transform
Schema Set	Core Components (UN/CEFACT) - NDR 3.0
Transform Rule Script	Schema Composer.BDTTransformRule ...
Qualifier	
BDTLibrary	BDTLibrary
BIELibrary	BIELibrary

The script can be written in either JavaScript, JScript or VBScript, and only has to implement this function (described here in JavaScript notation):

```
function TranslateStereotype(srcStereo)
{
    var destStereo = srcStereo
    if (srcStereo == "BDT")
    {
        destStereo = "My_BDT"
    }
    return destStereo;
}
```

MDG Technologies - UML Profile Extensions

The Schema Composer works with MDG technologies. The standards it uses for schema generation, *other than Generic,* are only meaningful for models that adhere to that framework. However, it is quite easy to extend an existing MDG Technology. Make sure that elements authored in your business specific domain or sub-domain provide consistently named metadata or 'Tagged Values'.

The Schema Composer supports extensions to UML profiles / frameworks through its scripting integration. When a script is assigned in the Schema Composer, the transform process will invoke this script and ask it to translate keywords. These keywords are usually UML stereotypes. If a particular technology is associated with the profile, the Schema Composer will invoke this function, passing it the name of the MDG Technology.

The script can return the input name, and no mapping will take place, or it can return the name of another MDG Technology. When this occurs, the Schema Composer will again ask for the function to optionally map any UML profiles. Finally it will ask the script to translate the stereotypes from the core technology.

The result of the model transform would then be that any UML elements of the sub model will show the extended Tagged Values in addition to any core Tagged Values.

Example script that maps MDG Technology

```
function TranslateStereotype (stereo)
{
  var newStereo = stereo;
  if (stereo == "UPCC3")
  {
      newStereo = "XXX UPCC3"
  }
  return newStereo;
}
```

Example script that maps UML profile

```
function TranslateStereotype (stereo)
{
  var newStereo = stereo;
  if (stereo == "UPCC3 - BIE Library Abstract Syntax")
  {
      newStereo = "UPCC3 - BIE Library XXX Syntax"
  }
  return newStereo;
}
```

Example script that maps UML Stereotype

```
function TranslateStereotype (stereo)
{
  var newStereo = stereo;
  if (stereo == "ABIE")
  {
      newStereo = "XXX ABIE";
  }
  return newStereo;
}
```

XSD Models

XML Schema Definition (XSD), also known as XML Schema, is a World Wide Web Consortium (W3C) XML technology that is used to specify the rules to which an XML document must adhere. XSD support is critical for the development of a complete Service Oriented Architecture (SOA), and the coupling of UML 2.5 and XML provides the natural mechanism for specifying, constructing and deploying XML-based SOA artifacts within an organization.

The UML Profile for XSD specifies a set of stereotypes, Tagged Values and constraints that can be applied to the UML model in order to change specific aspects of the resulting schema. Enterprise Architect provides native support for the XSD Profile through the XML Schema page of the Diagram Toolbox. The XSD Profile supported by Enterprise Architect is an adaptation of the profile defined in the publication Modeling XML Applications with UML.

Working with the XSD Profile through Enterprise Architect, you can rapidly model, forward engineer and reverse engineer XML Schema.

You can also quickly define and generate XSD and other schema using the Enterprise Architect Schema Composer.

Modeling XSD

You can model XML schemas at two levels, using UML Class diagrams that:

- Have no XML schema-specific implementation details, to be generated directly by Enterprise Architect's Schema Generator; the generator applies a set of default mappings to convert the abstract model Package to a W3C XML Schema (XSD) file
- Are refined with XML schema-specific definitions using the 'XML Schema' pages of the Diagram Toolbox, which provides the structures of the UML profile for XSD

Model an XML Schema

Step	Action
1	In the Browser window, create the top-level project structure you need (Model and Views), and click on the appropriate View.
2	Click on the 'New Package' option in the Browser window header drop-down menu. The 'New Model Package' dialog displays.
3	In the 'Name' field type the name of the new Package, and select the 'Create diagram' radio

	button. Click on the OK button. The 'New Diagram' dialog displays.
4	In the 'Name' field type the name of the new diagram. In the 'Select From' panel select 'UML Structural', and in the 'Diagram Types' panel select 'Class'.
5	Click on the OK button. In the Browser window, double-click on the icon next to the new diagram's name; the diagram opens in the Diagram View, with the 'Class' pages displaying in the Diagram Toolbox. At this point you can either: • Create a Class diagram using the Class Toolbox icons, or • Create a tailored XML Schema diagram using the 'XML Schema' pages of the Diagram Toolbox (continue to step 6)
6	Click on to display the 'Find Toolbox Item' dialog and specify 'XML Schema' to display the 'XML Schema' Toolbox pages.
7	Click on the 'Schema' icon from the Toolbox and drag it into the Class diagram. The 'XSD schema Properties' dialog displays. Complete this dialog, and click on the OK button.

	The 'New Diagram' dialog displays.
8	Again, in the 'Name' field type the name of the new diagram. In the 'Select From' panel select 'UML Structural', and in the 'Diagram Types' panel select 'Class'. Click on the OK button.
9	An XSDschema stereotyped Package is created in the Browser window and on the diagram, with a child Class diagram. Double-click on the Package on the diagram to open the child Class diagram, and use the constructs from the XML Schema Toolbox to model the XML Schema.

Notes

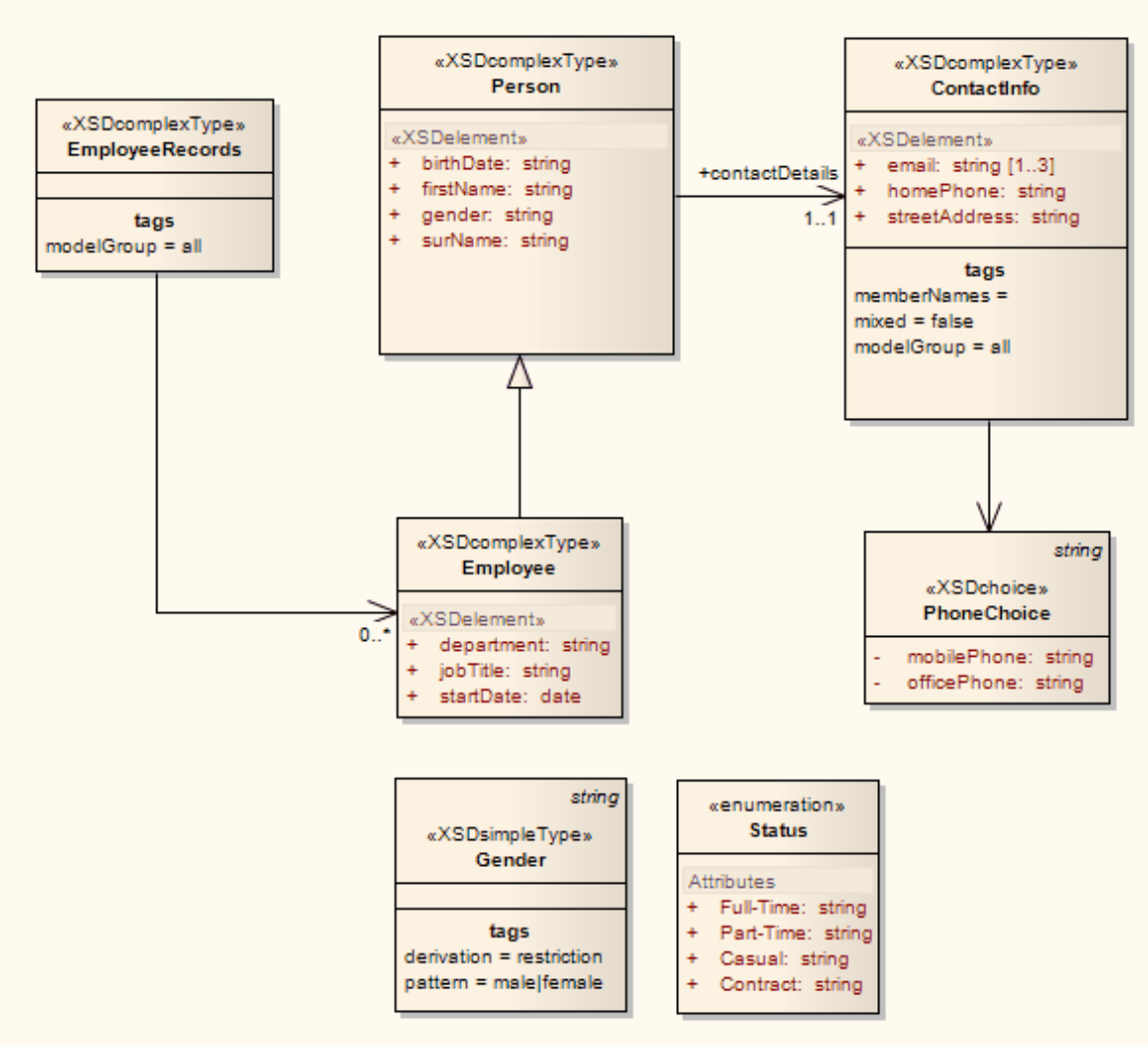
- The UML attributes of the Classes map directly to XML elements or attributes
- If you have modeled your XSD Schema as a straight Class diagram, you can define and generate schema from it using the Schema Composer
- Classes in an XML Schema model have no methods since there is no meaningful correspondence between Class methods and XSD constructs
- Modeling Restrictions - these XML Schema constructs

cannot be modeled in Enterprise Architect:

- appinfo
- field
- key
- keyref
- notation
- redefine
- selector
- substitutionGroup
- unique

XSD Diagrams

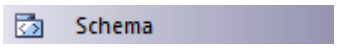
This example diagram shows a Class diagram containing XSD-specific elements created using the 'XSD Schema' pages of the Diagram Toolbox. The diagram models an employee records system.



Schema Package

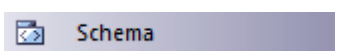
An «XSDschema» stereotyped Package acts as a container for the XSD constructs, from which XML Schema can be generated. All Classes in the Package are defined within one schema; the Schema element provides the default schema-wide settings. You can create an «XSDschema» Package by dragging the Schema icon from the XML Schema Toolbox and dropping it directly onto a diagram.

Toolbox Icon



Access

To display the XSD schema 'Properties' dialog for the selected «XSDschema» stereotyped Package, use one of the methods outlined here:

Ribbon	Design > Package > Manage > Properties
Context Menu	Right-click on «XSDschema» stereotyped Package | Properties
Other	In Browser window, double-click on existing «XSDschema» stereotyped Package, or Drag  icon from toolbox

	onto diagram (this creates a new «XSDschema» stereotyped Package)

Define Properties

Field/Button	Action
Schema Name	If you do not want to use the default name of the schema Package, overtype it with another name.
Target Namespace	(Optional) Type in the target namespace for this Schema Package.
Prefix	(Optional) Type in the abbreviated text to represent the Target Namespace.
Default Namespace	(Optional) Type in the default namespace for all non-prefixed XSDelements and XSDattributes.
Schema File	Type in or browse for (click on ...) the file path where the XML Schema file for this Package is to be generated.
XMLNS	Identify the additional namespace or namespace-prefix pairs used in this

	Schema Package. To add a namespace or namespace-prefix pair, click on the New button; to edit an existing entry, double-click on it. In either case, the 'Namespace Details' dialog displays. • Prefix - Type in the abbreviated text to represent the Namespace • Namespace - Type in the name of the Namespace • OK - Click on this button to save the new information and close the 'Namespace Details' dialog • Cancel - Click on this button to discard the new information and close the 'Namespace Details' dialog • Help - Click on this button to display this Help topic To remove an entry from the list, click on it and click on the Delete button.
OK	Click on this button to save the schema data entered and close the XSD schema 'Properties' dialog.
Cancel	Click on this button to discard the schema data entered and close the XSD schema 'Properties' dialog.

Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing Schema Package information. Click on the button to open the UML element 'Properties' dialog for the Schema element.

Notes

- The default schema-wide settings are defined by Tagged Values, which you can review on the 'Tags' tab of the schema element 'Properties' dialog, or the Properties window for the element; you can edit the schema-wide settings if you need to, or provide element-specific overrides in the properties and Tagged Values of the individual XSD construct elements

Global Element

An «XSDtopLevelElement» stereotyped Class acts as a XSD global element. You can create it by dragging the 'Element' icon from the XML Schema Toolbox and dropping it directly onto a diagram.

Toolbox Icon

Access

To display the 'XSD element Properties' dialog for the selected «XSDtopLevelElement» stereotyped Class, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties
Context Menu	Right-click on «XSDtopLevelElement» stereotyped Class | Properties
Keyboard Shortcuts	Alt+Enter
Other	In Browser window, double-click on «XSDtopLevelElement» stereotyped Class, or Drag icon from toolbox

	onto diagram (this creates a new «XSDtopLevelElement» stereotyped Class)

Define Properties

Field/Button	Action
Name	If you do not want to use the default name of the global element, overtype it with another name.
Type	Either: • Type the name of a data type, or • Click on the drop-down arrow and select an XSD built-in dataType from the list, or • Click on the ... button and browse for an existing XSD classifier element, or • Select one of these two checkboxes
Nested complexType	Select this checkbox to create an XSDcomplexType as a child of this global element.
Nested	Select this checkbox to create an

simpleType	XSDsimpleType as a child of this global element.
Value	(Optional) If you have entered an XSD built-in data type in the 'Type' field, type in a value.
Default	Select this radio button to set the Value as a default value.
Fixed	Select this radio button to set the Value as a fixed value.
Annotation	(Optional) Type in any notes you need for this element.
OK	Click on this button to save the element data entered and close the XSD element 'Properties' dialog.
Cancel	Click on this button to discard the element data entered and close the XSD element 'Properties' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing XSD element

	information. Click on the button to open the UML element 'Properties' dialog for the global element.

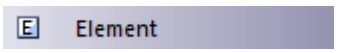
Notes

- The fields 'Type', 'Nested complexType' and 'Nested simpleType' are mutually exclusive; selecting one disables the others
- The fields 'Nested complexType' and 'Nested simpleType' are available in the dialog only when creating a new global element (and not when editing the global element)
- A Global element:
  - Cannot contain any UML attributes
  - Cannot be the source of an Association connector
  - Can be the target of an Association connector from a Complex Type Class or Group Class element
  - Cannot be the target of a Generalization connector
  - Can be the source of one Generalization connector to a Complex Type Class or Simple Type Class

Local Element

A Local element is an «XSDelement» stereotyped UML attribute that acts as a local XSD element. You can create it by dragging the 'Element' icon from the XML Schema Toolbox and dropping it onto an «XSDcomplexType» or «XSDgroup» stereotyped Class.

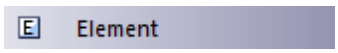
Toolbox Icon



Access

To display the 'XSD element Properties' dialog for the selected «XSDelement» stereotyped UML attribute, use one of the methods outlined here.

Ribbon	With a specific ?XSDelement? stereotyped attribute selected in a diagram: Design > Element > Features > Attributes
Context Menu	With a specific ?XSDelement? stereotyped attribute selected in a diagram : Right-click on attribute | View Properties
Keyboard	With a specific ?XSDelement?

Shortcuts	stereotyped attribute selected in a diagram : F9
Other	Double-click on ?XSDelement? stereotyped attribute, or Drag  icon onto ?XSDcomplexType? or ?XSDgroup? stereotyped Class, (this creates a new ?XSDelement? within the Class)

Define Properties

Field/Button	Action
Name	If you do not want to use the default name of the local element, overtype it with another name.
Type	Either: • Type the name of a data type, or • Click on the drop-down arrow and select an XSD built-in dataType from the list, or • Click on the ... button and browse for an existing XSDcomplexType or

	XSDsimpleType element as a classifier
Reference	(Optional) Specify whether to use the ref attribute (instead of the type attribute) to refer to the XSDcomplexType or XSDsimpleType element you selected in the 'Type' field, in the generated XSD.
Value	(Optional) If you have entered an XSD built-in data type in the 'Type' field, type in a value.
Default	Select this radio button to set the Value as a default value.
Fixed	Select this radio button to set the Value as a fixed value.
MinOccurs	(Optional) Type the minimum number of times this element must occur in the Class. Type '0' to indicate that the element is optional. The default value is '1'.
MaxOccurs	(Optional) Type the maximum number of times this element can occur in the Class. Type unbounded to indicate that there is no limit to the number of times the

	element can occur. The default value is 1.
Form	(Optional) Click on the drop-down arrow and select whether or not to qualify the element: • qualified - Use the Prefix defined in the Schema Package to qualify this element • unqualified - Do not qualify this element
Annotation	(Optional) Type in any notes you need for this local element.
OK	Click on this button to save the element data entered and close the XSD element 'Properties' dialog.
Cancel	Click on this button to discard the element data entered and close the XSD element 'Properties' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing XSD element

	information. Click on the button to open the attribute properties for the local element.

Notes

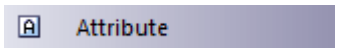
- Only ?Complex Type?, ?Group? and ?Model Group? stereotyped elements can have this UML Attribute

Global Attribute

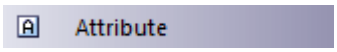
A Global Attribute is an «XSDtopLevelAttribute» stereotyped Class. You can create it by dragging the 'Attribute' icon from the XML Schema Toolbox and dropping it directly onto a diagram.

Access

To display the 'XSD attribute Properties' dialog for the selected «XSDtopLevelAttribute» stereotyped element, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties
Context Menu	Right-click on «XSDtopLevelAttribute» stereotyped element| Properties
Keyboard Shortcuts	Alt+Enter
Other	Double-click on ?XSDtopLevelAttribute? stereotyped Class, or Drag  icon from toolbox and drop directly onto the diagram (this creates a new ?XSDtopLevelAttribute? stereotyped Class)

Toolbox Icon



Define Properties

Field/Button	Action
Name	If you do not want to use the default name of the global attribute, overtype it with another name.
Type	Either: • Type the name of a data type, or • Click on the drop-down arrow and select an XSD built-in dataType from the list, or • Click on the ... button and browse for an existing XSDsimpleType element as a classifier Alternatively, select the 'Nested simpleType' checkbox.
Nested simpleType	Select this checkbox to create an XSDsimpleType element as a child of this global attribute element.

Value	(Optional) If you have selected an XSD built-in dataType in the 'Type' field, type in a value.
Default	Select this radio button to set the 'Value' field as a default value.
Fixed	Select this radio button to set the 'Value' field as a fixed value.
Form	(Optional) Click on the drop-down arrow and select: • qualified to use any Prefix supplied on the Schema Package to qualify this attribute, or • unqualified to show no qualifying prefix on the attribute name
Annotation	(Optional) Type in any notes you need for this attribute.
OK	Click on this button to save the attribute data entered and close the XSD attribute 'Properties' dialog.
Cancel	Click on this button to discard the attribute data entered and close the XSD attribute 'Properties' dialog.

Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing XSD element information. Click on the button to open the UML element 'Properties' dialog for the global attribute Class.

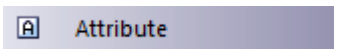
Notes

- The field 'Nested simpleType' is available in the dialog only when creating a new global attribute (and not when editing the global attribute)
- The fields 'Type' and 'Nested simpleType' are mutually exclusive; selecting one disables the other
- A Global attribute:
  - Cannot contain any UML attributes
  - Cannot be the source of an Association connector
  - Can be the target of an Association connector from a Complex Type Class
  - Cannot be the target of a Generalization connector
  - Can be the source of one Generalization connector to a Simple Type Class

Local Attribute

A local attribute is an «XSDattribute» stereotyped UML attribute. You can create it by dragging the 'Attribute' icon from the XML Schema Toolbox and dropping it onto an «XSDcomplexType» or «XSDattributeGroup» stereotyped Class.

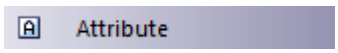
Toolbox Icon



Access

To display the 'XSD attribute Properties' dialog for the selected «XSDattribute» stereotyped UML attribute, use one of the methods outlined here.

Ribbon	With a specific ?XSDattribute? stereotyped UML attribute selected on a diagram: Design > Element > Features > Attributes
Context Menu	With a specific ?XSDattribute? stereotyped UML attribute selected on a diagram: Right-click on attribute | View Properties
Keyboard	With a specific ?XSDattribute?

Shortcuts	stereotyped UML attribute selected on a diagram: F9
Other	Double-click on the ?XSDattribute? stereotyped UML attribute, or Drag  icon from the Toolbox and drop it onto an ?XSDcomplexType? or ?XSDattributeGroup? stereotyped Class (this creates a new ?XSDattribute? stereotyped UML attribute)

Define Properties

Field/Button	Action
Name	If you do not want to use the default name of the local attribute, overtype it with another name.
Type	Either: • Type the name of a data type, or • Click on the drop-down arrow and select an XSD built-in dataType from the list, or

	• Click on the ... button and browse for an existing XSDsimpleType element as a classifier
Reference	(Optional) Specify whether to use the ref attribute (instead of the type attribute) to refer to the XSDsimpleType element you selected in the 'Type' field, in the generated XSD.
Value	(Optional) If you have entered an XSD built-in data type in the 'Type' field, type in a value.
Default	Select this radio button to set the Value as a default value.
Fixed	Select this radio button to set the Value as a fixed value.
Form	(Optional) Click on the drop-down arrow and select whether or not to qualify the attribute: • qualified - Use the Prefix defined in the Schema Package to qualify this attribute • unqualified - Do not qualify this attribute

Annotation	(Optional) Type in any notes you need for this local attribute.
OK	Click on this button to save the attribute data entered and close the XSD attribute 'Properties' dialog.
Cancel	Click on this button to discard the element data entered and close the XSD attribute 'Properties' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing XSD attribute information. Click on the button to open the attribute properties for the local attribute.

Notes

- Only Complex Types and Attribute Groups can have this UML attribute

Attribute Group

An Attribute Group Class is used to group a set of «XSDattribute» stereotyped UML attributes and Simple Type Classes that can be referenced from an «XSDcomplexType» stereotyped Class. You can create it by dragging the 'Attribute Group' icon from the XML Schema Toolbox and dropping it directly onto a diagram.

Toolbox Icon

Access

To display the 'XSD Attribute Group Properties' dialog for the selected «XSDattributeGroup» stereotyped Class, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties
Context Menu	Right-click on «XSDattributeGroup» stereotyped Class | Properties
Keyboard Shortcuts	Alt+Enter
Other	Double-click on ?XSDattributeGroup? stereotyped Class, or

	Drag icon from toolbox onto diagram (this creates a new ?XSDattributeGroup? stereotyped Class)

Define Properties

Field/Button	Action
Name	If you do not want to use the default name of the Attribute Group, overtype it with another name.
Annotation	(Optional) Type in any notes you need for this Attribute Group.
OK	Click on this button to save the attribute group data entered and close the XSD Attribute Group 'Properties' dialog.
Cancel	Click on this button to discard the attribute group data entered and close the XSD Attribute Group 'Properties' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are

	editing existing XSD attribute group information. Click on the button to open the UML element 'Properties' dialog for the attribute group.

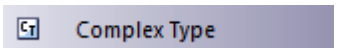
Notes

- An Attribute Group element:
  - Cannot be the child of any other XSD Class
  - Can contain only «XSDattribute» stereotyped UML attributes and Simple Type Classes
  - Can be the source of an Association connector to another Attribute Group
  - Can be the target of an Association connector from a Complex Type Class
  - Cannot be the source or target of a Generalization connector

Complex Type

An «XSDcomplexType» stereotype is applied to a generic UML Class, to tailor the generation of a complexType definition in the Schema. You can create an «XSDcomplexType» stereotyped Class by dragging the Complex Type icon from the XML Schema Toolbox and dropping it directly onto a diagram.

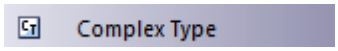
Toolbox Icon



Access

To display the 'XSD complexType Properties' dialog for the selected «XSDcomplexType» stereotyped Class, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties
Context Menu	Right-click on «XSDcomplexType» stereotyped Class | Properties
Keyboard Shortcuts	Alt+Enter
Other	Double-click on ?XSDcomplexType? stereotyped Class, or

	Drag  icon from toolbox onto diagram (this creates a new ?XSDcomplexType? stereotyped Class)

Define Properties

Field/Button	Action
Name	If you do not want to use the default name of the complexType Class, overtype it with another name.
Model Group	Click on the down-arrow and select the option that defines how the child elements of this complexType should occur in the Schema. • 'sequence' - the child elements must occur in the specified order • 'choice' - only one of the child elements can occur • 'all' - the child elements can occur in any order
MinOccurs	(Optional) Type the minimum number of times this element must occur in the Class.

	Type 0 to indicate that the element is optional. The default value is 1.
MaxOccurs	(Optional) Type the maximum number of times this element can occur in the Class. Type unbounded to indicate that there is no limit to the number of times the element can occur. The default value is 1.
Annotation	(Optional) Type any notes you need for this element.
Abstract	(Optional) Select this checkbox to use this complexType in an instance XML file.
Mixed	(Optional) Select this checkbox to allow character data to display among the child elements.
OK	Click on this button to save the complexType data entered and close the XSD complexType 'Properties' dialog.
Cancel	Click on this button to discard the complexType data entered and close the

	XSD complexType 'Properties' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing XSD complexType information. Click on the button to open the UML element 'Properties' dialog for the complexType Class.

Notes

- A complexType can:
  - Contain both XSDelement and XSDattribute stereotyped UML attributes
  - Contain other complexTypes as child elements
  - Be a child of a Global Element
  - Be the source of Association connectors to other complexTypes, Simple Types, Attribute Groups, Groups and Model Groups
  - Be the source of a maximum of one Generalization connector to either another complexType or a Simple Type Class

Simple Type

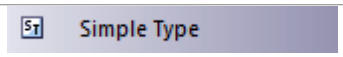
An «XSDsimpleType» stereotype is applied to a generic UML Class, to tailor the generation of a simpleType definition in the Schema. You can create an «XSDsimpleType» Class by dragging the Simple Type icon from the XML Schema Toolbox and dropping it directly onto a diagram.

Toolbox Icon

Access

To display the 'XSD simpleType Properties' dialog for the selected «XSDsimpleType» stereotyped Class, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties
Context Menu	Right-click on «XSDsimpleType» stereotyped Class | Properties
Keyboard Shortcuts	Alt+Enter
Other	Double-click on ?XSDsimpleType? stereotyped Class, or

	Drag  icon from toolbox onto diagram (this creates a new ?XSDsimpleType? stereotyped Class)

Define Properties

Field/Button	Action
Name	If you do not want to use the default name of the simpleType element, overtype it with another name.
Type	Either: • Type the name of a data type, or • Click on the drop-down arrow and select an XSD built-in dataType from the list, or • Click on the button and browse for an existing ?XSDsimpleType? element as a classifier
Restriction	Select this radio button to restrict the value of this simpleType to that of the selected Type. The various restrictions (facets) on the simpleType are available as Tagged

	Values on this Class.
List	Select this radio button to specify this simpleType as a list of values of the selected Type.
Annotation	(Optional) Type any notes you need for this element.
OK	Click on this button to save the simpleType data entered and close the XSD simpleType 'Properties' dialog.
Cancel	Click on this button to discard the simpleType data entered and close the XSD simpleType 'Properties' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing XSD simpleType information. Click on the button to open the UML element 'Properties' dialog for the simpleType Class.

Notes

- A simpleType:
- Cannot contain any ?XSDelement? or ?XSDattribute? stereotyped UML attributes
- Cannot contain any child Classes
- Cannot be the source of an Association connector
- Can be the target of a Generalization connector
- Can have at the most one Generalization connector to another simpleType Class

Group

The Group Class is used to group a set of «XSDelement» stereotyped UML attributes, Complex Type Classes and Simple Type Classes that can be referenced from an «XSDcomplexType» Class. You can create this type of element by dragging the Group icon from the XML Schema Toolbox and dropping it directly onto a diagram.

Toolbox Icon

Access

To display the 'XSD group Properties' dialog for the selected «XSDgroup» stereotyped Class, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties
Context Menu	Right-click on «XSDgroup» stereotyped Class | Properties
Keyboard Shortcuts	Alt+Enter
Other	Double-click on ?XSDgroup? stereotyped Class, or

	Drag icon from toolbox onto diagram (this creates a new ?XSDgroup? stereotyped Class)

Define Properties

Field/Button	Action
Name	If you do not want to use the default name of the Group element, overtype it with another name.
Model Group	Click on the drop-down arrow and select the value that defines how the child elements of this group should occur in the Complex Type Class: • sequence - to specify that the child elements must occur in the specified order • choice - to specify that only one of the child elements can occur • all - to specify that the child elements can occur in any order
Annotation	(Optional) Type any notes you need for this element.

OK	Click on this button to save the Group data entered and close the XSD group 'Properties' dialog.
Cancel	Click on this button to discard the Group data entered and close the XSD group 'Properties' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing XSD Group information. Click on the button to open the UML element 'Properties' dialog for the Group Class.

Notes

- A Group element can:
- Contain only ?XSDelement? stereotyped UML attributes
- Contain Complex Types and Simple Types as child elements
- Be the source of Association connectors to other Complex Types, Simple Types and Groups
- Be the target of an Association connector from a Complex

Type element

- Not be the source or target of a Generalization connector

Any

An «XSDany» stereotyped Class allows a Complex Type Class to contain elements that are not specified in the Schema Package. You can create it by dragging the Any icon from the XML Schema Toolbox and dropping it directly onto a diagram.

Toolbox Icon

Access

To display the 'XSD any Properties' dialog for the selected «XSDany» stereotyped Class, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties
Context Menu	Right-click on «XSDany» stereotyped Class | Properties
Keyboard Shortcuts	Alt+Enter
Other	Double-click on ?XSDany? stereotyped Class, or Drag icon from toolbox

	onto diagram (this creates a new ?XSDany? stereotyped Class)

Define Properties

Field/Button	Action
Name	If you do not want to use the default name of the Any element, overtype it with another name.
Namespace	(Optional) Type the namespace to contain the elements that can be used in the Complex Type.
ProcessContents	(Optional) Click on the drop-down arrow and select the value that defines how the XML Parser should validate these elements: • lax - to attempt to validate the elements against their Schema; no error is flagged when the Schema cannot be obtained • skip - to skip validating the elements • strict - to validate the elements against their Schema and flag an error if the

	Schema is unobtainable
Annotation	(Optional) Type any notes you need for this element.
OK	Click on this button to save the 'Any' element data entered and close the XSD Any 'Properties' dialog.
Cancel	Click on this button to discard the Any element data entered and close the XSD group 'Properties' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing ?XSDany? element information. Click on the button to open the UML element 'Properties' dialog for the Any Class.

Notes

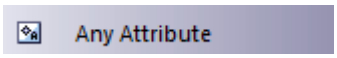
- An Any Class:
- Cannot contain any UML Attributes or child XSD Classes

- Cannot be the child of any XSD Class
- Cannot be the target of a Generalization connector
- Cannot be the source of an Association or Generalization connector
- Can be the target of Association connectors from Complex Types, Groups and Model Groups
- Must be the target of at least one incoming Association connector

Any Attribute

The «XSDany» stereotyped UML attribute allows a Complex Type element or an Attribute Group element to contain attributes that are not specified in the Schema Package. You can create it by dragging the 'Any Attribute' icon from the XML Schema Toolbox and dropping it onto an «XSDcomplexType» or «XSDattributeGroup» stereotyped Class.

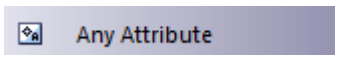
Toolbox Icon



Access

To display the 'XSD anyAttribute Properties' dialog for the selected «XSDany» stereotyped UML attribute, use one of the methods outlined here.

Ribbon	With a specific ?XSDany? stereotyped UML attribute selected on a diagram: Design > Element > Features > Attributes
Context Menu	With a specific ?XSDany? stereotyped UML attribute selected on a diagram: Right-click on attribute | View Properties
Keyboard	With a specific ?XSDany? stereotyped

Shortcuts	UML attribute selected on a diagram: F9
Other	Double-click on the ?XSDany? stereotyped UML attribute, or Drag  icon from the Toolbox and drop it onto an ?XSDcomplexType? or ?XSDattributeGroup? stereotyped Class (this creates a new ?XSDany? stereotyped UML attribute)

Define Properties

Field/Button	Action
Name	If you do not want to use the default name of the attribute, overtype it with another name.
Namespace	(Optional) Type the namespace to contain the attributes that can be used in the Complex Type or Attribute Group elements.
ProcessConte	(Optional) Click on the drop-down arrow and select the value that defines how the

nts	XML Parser should validate these attributes: • lax - to attempt to validate the attributes against their Schema; no error is flagged when the Schema cannot be obtained • skip - to skip validating the attributes • strict - to validate the attributes against their Schema and flag an error if the Schema is unobtainable
Annotation	(Optional) Type any notes you need for this attribute.
OK	Click on this button to save the attribute data entered and close the XSD anyAttribute 'Properties' dialog.
Cancel	Click on this button to discard the attribute data entered and close the XSD anyAttribute 'Properties' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing ?XSDany? attribute information. Click on the button to open the attribute

	properties for the ?XSDany? attribute.

Notes

- Only Complex Type and Attribute Group elements can have this UML attribute

Union

A Union Class is a Simple Type element that defines a collection of Simple Types. You can create it by dragging the Union icon from the XML Schema Toolbox and dropping it directly on a diagram.

Toolbox Icon

Access

To display the 'XSD union Properties' dialog for the selected «XSDunion» stereotyped Class, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties
Context Menu	Right-click on «XSDunion» stereotyped Class | Properties
Keyboard Shortcuts	Alt+Enter
Other	Double-click on ?XSDunion? stereotyped Class, or Drag Union icon from toolbox onto diagram (this creates a new

	?XSDunion? stereotyped Class)

Define Properties

Field/Button	Action
Name	If you do not want to use the default name of the Union, overtype it with another name.
Member Types	Click on the ... button to display the 'XSD Union Members' dialog, and select built-in XSD datatypes and Simple Type elements to be members of the collection. • Choose - Instead of typing or selecting values in the 'Class Name' field, click on this button to display the 'Select Classifier' browser and locate and select a Simple Type element; click on the OK button to close the browser and immediately add the selected element to the 'Type Details' list This option is generally used to specify objects that are in the same Package as the Union element, but you can select objects in any other Package also • Add - Click on this button to add the

	data type or element specified in the 'Class Name' field to the 'Type Details' list • Accept classifier even if not in model - Select this checkbox to include elements or data types that have been named but that are not present in the same model Package as the Union element • Type Details - Review the list of selected elements or data types; if you intend to remove an object from the list, highlight it and click on the Delete Selected button • Delete Selected - Click on this button to remove the currently-selected Classifier from the 'Type Details' list • Close - Click on this button to close the 'XSD Union Members' dialog and to list the selected elements and data types in the 'Member Types' field
Annotation	(Optional) Type any notes you need for this element.
OK	Click on this button to save the attribute data entered and close the XSD union 'Properties' dialog.

Cancel	Click on this button to discard the attribute data entered and close the XSD union 'Properties' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing ?XSDunion? element information. Click on the button to open the UML element 'Properties' dialog for the ?XSDunion? element.

Notes

- When you click on the Close button on the XSD union 'Properties' dialog, a Generalization connector is added to the diagram from the XSD Union element to each of the member elements on the same diagram; any elements not on the same diagram are listed in the top right corner of the XSD Union element
- If the Member Types that are not on the same diagram as the Union element are not listed, select 'Start > Appearance > Preferences > Preferences > Diagram > Behavior' and select the 'Show Hidden Parents' checkbox
- A Union element:

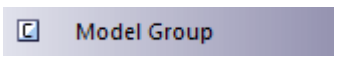
- Cannot contain any child Classes
- Cannot contain any «XSDelement» or «XSDattribute» stereotyped UML attributes
- Cannot be the source of an Association connector
- Can be the target of an Association connector from a Complex Type element
- Can be the target of a Generalization connector from a Simple Type element

Model Group

You can create an «XSDsequence», «XSDchoice» or «XSDall» stereotyped Class by dragging the Model Group icon from the XML Schema Toolbox and dropping it directly onto a diagram.

An «XSDsequence» model group (the default model group type) is a container for the attributes and associations owned by the Class. The model group is in turn added to the model groups of the Class's respective owners. Tagged Values specified by owners of the Class persist through to the child elements of the model group; if memberNames are unqualified for a complexType, so are the children of this model group when added to that complexType.

Toolbox Icon



Access

To display the 'XSD Model Group Properties' dialog for the selected «XSDsequence», «XSDchoice» or «XSDall» stereotyped Class, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties
Context Menu	Right-click on «XSDsequence», «XSDchoice» or «XSDall» stereotyped

	Class | Properties
Keyboard Shortcuts	Alt+Enter
Other	• Double-click on ?XSDsequence?, ?XSDchoice? or ?XSDall? stereotyped Class, or • Drag the icon from the Toolbox onto the diagram (this creates a new Model Group element; you can choose from the ?XSDsequence?, ?XSDchoice? or ?XSDall? stereotypes, of which ?XSDsequence? is the default)

Define Properties

Field/Button	Action
Name	If you do not want to use the default name of the Model Group, overtype it with another name.
Model Group	Click on the drop-down arrow and select the value that defines how the child elements of this group should occur in the

	Complex Type Class: • sequence - to specify that the child elements must occur in the specified order; creates an ?XSDsequence? stereotyped Class • choice - to specify that only one of the child elements can occur; creates an ?XSDchoice? stereotyped Class • all - to specify that the child elements can occur in any order; creates an ?XSDall? stereotyped Class
MinOccurs	(Optional) Type the minimum number of times this element must occur in the Class. Type 0 to indicate that the element is optional. The default value is 1.
MaxOccurs	(Optional) Type the maximum number of times this element can occur in the Class. Type unbounded to indicate that there is no limit to the number of times the element can occur. The default value is 1.
Annotation	(Optional) Type any notes you need for this element.

OK	Click on this button to save the Model Group data entered and close the XSD element 'Properties' dialog.
Cancel	Click on this button to discard the Model Group data entered and close the XSD element 'Properties' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing Model Group element information. Click on the button to open the UML element 'Properties' dialog for the Model Group Class.

Notes

- A Model Group:
- Can contain only ?XSDelement? stereotyped UML attributes
- Can contain Complex Types and Simple Types as child elements
- Can be the source of Association connectors to Complex

Type, Simple Type, Group and Model Group elements

- Must be the target of least one incoming Association connector from a Complex Type
- Cannot be the source or target of a Generalization connector

Enumeration

An Enumeration defines a list of acceptable values for the Class. You can create an Enumeration element by dragging the Enum icon from the XML Schema Toolbox and dropping it directly onto a diagram.

Toolbox Icon

Access

To display the 'XSD enumeration Properties' dialog for the selected «enumeration» stereotyped element, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties
Context Menu	Right-click on «enumeration» stereotyped element | Properties
Keyboard Shortcuts	Alt+Enter
Other	Double-click on ?enumeration? stereotyped element, or Drag Enum icon from toolbox and drop directly onto the diagram (this creates a

	new ?enumeration? stereotyped element)

Define Properties

Field/Button	Action
Name	If you do not want to use the default name of the Enumeration, overtype it with another name.
Type	Either: • Type the name of a data type, or • Click on the drop-down arrow and select an XSD built-in dataType from the list, or • Click on the ... button and browse for an existing XSDsimpleType element
Values	Type each of the values, separated by commas, for the selected Type. These values are listed on the element as attributes.
Annotation	(Optional) Type any notes you need for this element.

OK	Click on this button to save the Enumeration element data entered and close the XSD enumeration 'Properties' dialog.
Cancel	Click on this button to discard the Enumeration element data entered and close the XSD enumeration 'Properties' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing Enumeration element information. Click on the button to open the UML element 'Properties' dialog for the Enumeration Class.

Notes

- An Enumeration:
- Cannot contain any ?XSDelement? or ?XSDattribute? stereotyped UML attributes
- Cannot contain any child Classes
- Cannot be the source of an Association connector

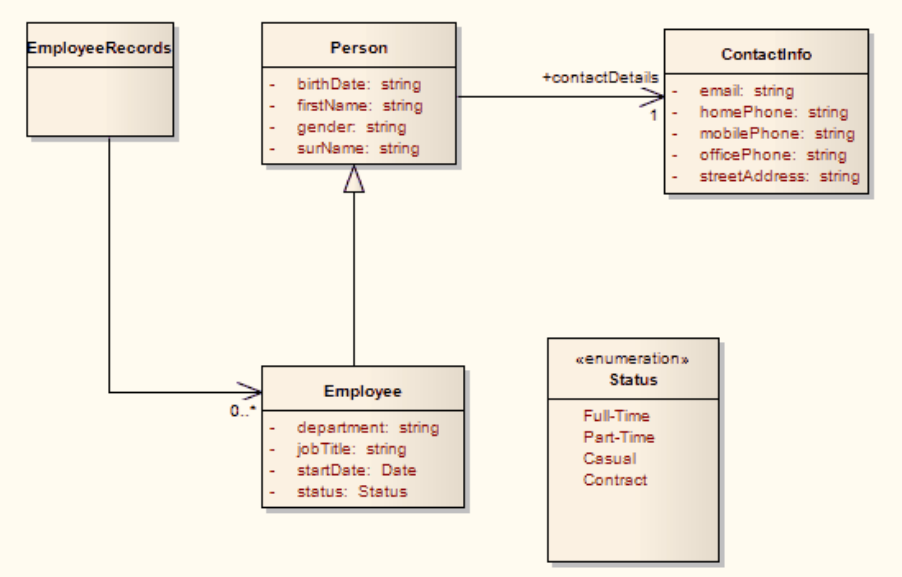
- Can be the target of a Generalization connector
- Can have at most one Generalization connector to a Simple Type Class

XML from Abstract Class Models

You can model XML schemas using only simple, abstract Class models. This makes it possible for an architect, for example, to start working at a higher level of abstraction without concern for the implementation details of a Schema. Whilst such an abstract model can subsequently be refined using the 'XML Schema' pages of the Toolbox, it can be also be generated directly by Enterprise Architect's Schema Generator, in which case the Schema Generator applies a set of default mappings to convert the abstract model to an XSD file.

Example

Structure	Detail
Diagram	This is a simple Class element version of the earlier Employee Details example model. It does not use XSD-specific stereotypes or Tagged Values.



Schema

This schema fragment can be generated from the example model:

```
<?xml version="1.0"?>
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:simpleType name="Status">
<xs:restriction base="xs:string">
<xs:enumeration value="Full-Time"/>
<xs:enumeration value="Part-Time"/>
<xs:enumeration value="Casual"/>
<xs:enumeration value="Contract"/>
</xs:restriction>
</xs:simpleType>
<xs:element name="Person"
type="Person"/>
```

```
<xs:complexType name="Person">
<xs:sequence>
<xs:element name="firstName"
type="xs:string"/>
<xs:element name="surName"
type="xs:string"/>
<xs:element name="birthDate"
type="xs:string"/>
<xs:element name="gender"
type="xs:string"/>
<xs:element name="contactDetails"
type="ContactInfo"/>
</xs:sequence>
</xs:complexType>
<xs:element name="Employee"
type="Employee"/>
<xs:complexType name="Employee">
<xs:complexContent>
<xs:extension base="Person">
<xs:sequence>
<xs:element name="status"
type="Status"/>
<xs:element name="jobTitle"
type="xs:string"/>
<xs:element name="startDate"
type="xs:date"/>
<xs:element name="department"
```

```
type="xs:string"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
<xs:element name="EmployeeRecords"
type="EmployeeRecords"/>
<xs:complexType
name="EmployeeRecords">
<xs:sequence>
<xs:element name="Employee"
type="Employee" minOccurs="0"
maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
<xs:element name="ContactInfo"
type="ContactInfo"/>
<xs:complexType name="ContactInfo">
<xs:sequence>
<xs:element name="homePhone"
type="xs:string"/>
<xs:element name="mobilePhone"
type="xs:string"/>
<xs:element name="officePhone"
type="xs:string"/>
<xs:element name="email"
type="xs:string"/>
```

	<xs:element name="streetAddress" type="xs:string"/> </xs:sequence> </xs:complexType> </xs:schema>

Default UML to XSD Mappings

When you are defining simple schemas using abstract Class models, the Enterprise Architect Schema Generator translates the UML information to XSD using a default mapping of UML to XSD constructs. These defaults are also used by the Schema Generator to generate unstereotyped elements in an abstract model.

When you model XML Schema using the 'XML Schema' pages of the Diagram Toolbox, the stereotypes and Tagged Values of the Toolbox elements override the default mappings.

Constructs

UML Construct	Default XSD Production Rules
Package	A Schema element is generated for the target Package. If the target Package includes Classes from another Package, which has the Tagged Values targetNamespace and targetNamespacePrefix set, these are included as attributes of the Schema element. In addition, an import or include element is created for each referenced Package:

	• An include element is used if the external Package shares the same targetNamespace Tagged Value as the target Package • An import element is used where the targetNamespaces differ
Class	A root-level element declaration and complexType definition are generated. The element name and type are the same as the Class name. An XSD sequence Model Group is also generated, to contain UML attributes generated as elements.
Attribute	An element is declared for each Class attribute. The element name is set to that of the UML attribute name. This is prefixed with the Class name to make the element unique. The minOccurs and maxOccurs attributes are set to reflect the attribute cardinality. If the attribute refers to another Class, the element declaration is followed by a complexType definition, which contains a reference to the appropriate complexType.
Association	An element is declared for each Association owned by a Class. The

	element name is set to that of the Association role. The minOccurs and maxOccurs attributes reflect the cardinality of the Association.
Generalization (Inheritance)	For single inheritances, an extension element is generated with the base attribute set to the base Class name. The UML attributes of the child Class are then appended to an XSDall Model Group within the extension element.
Enumeration	A simpleType element is declared for the Enumeration with the name attribute set to the Enumeration name. A Restriction element is generated with base set to string. Each of the Enumeration attributes is appended to the Restriction element as XSD Enumeration elements with value set to the UML attribute name. Any type specification for the UML attributes is ignored by the schema generator.

Notes

- If left unspecified, the minOccurs and maxOccurs attributes default to 1
- If the direction of the Association is unspecified, the

owner is assumed to be the source

Generate XSD

When you have developed your XML Schema model, either as an abstract Class model or a tailored XSD Class model, you can forward-engineer it into W3C XML Schema (XSD) files using the Generate XML Schema feature. As an XML Schema corresponds to a UML Package in Enterprise Architect, XML Schema generation is a Package-level operation.

You define the location of the file into which the XML Schema is to be generated, in the Schema Package element in your model.

Access

Ribbon	Develop > Schema Modeling > Export XSD

Generate Schema files

Option	Action
Encoding	Either: • Click on the drop-down arrow and select the XML encoding scheme to

	use, or • Click on the Default button to apply the default encoding scheme (UTF-8)
Generate global element for all global ComplexTypes ('Garden of Eden' style)	Selected by default to generate Schema in the Garden of Eden style, containing a global element. Clear the checkbox if you want to omit the global element.
Generate XSD for Referenced Packages	Select the checkbox to generate Schema for Packages that are referenced by any of the Packages selected on this dialog.
Prompt when missing Filename	Select the checkbox to prompt, during Schema generation, for a filename for a referenced Package if the path into which to generate the Schema file is missing. This option is not available if the 'Generate XSD for Referenced Packages' option is not selected.
Use relative-path to reference	Select the checkbox to use a relative-path in the XSD import (or XSD include) statement when referencing external

XSDs (if 'schemaLocation' tag is empty)	Packages, provided that the schemaLocation tag is empty on the referenced Packages. You set the 'Schema File' field on the 'XSD Schema Properties' dialog (the element 'Properties' dialog for a Schema element) for the referenced and referencing XSDschema stereotyped Packages, so that the relative-path is correctly determined.
Generate XSD for Child packages	Select the checkbox to generate schema for child Packages of the selected Package, and then select either: • Include all packages - to list all child Packages under the parent Package in the list box, or • Include <XSDschema> packages - to list only those Packages that have the stereotype ?XSDschema? The list-box shows, for each Package, the Package name and the file path into which the schema file can be generated (if set). To change the file path for a Package, double-click on the entry in the list-box and type in or browse for the new file path in the prompt field. If the Package has a filepath already set,

	its checkbox is selected by default, to generate an XSD schema; if you do not want to generate an XSD schema from that Package, you can deselect the checkbox. If you select the checkbox against a Package that does not have a filepath set, the prompt automatically displays for the filepath.
Generate	Click on this button to generate the Schema for each of the Packages selected in the list-box.
Close	Click on this button to close the dialog, without saving your option selections.
View Schema	Click on this button to view the generated Schema for a Package highlighted in the list-box.
Progress	Check the progress of Schema generation.

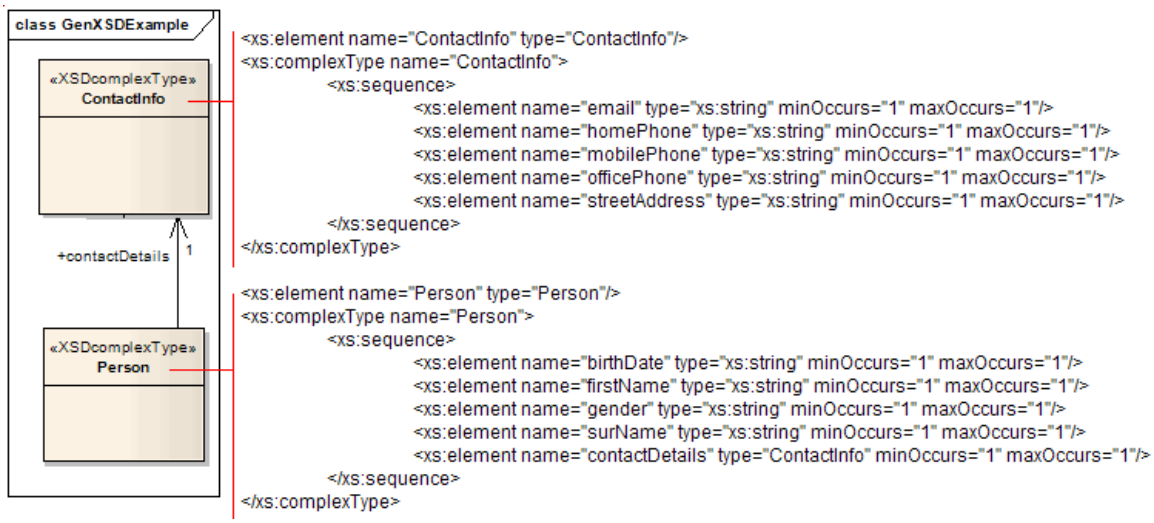
Generate Global Element

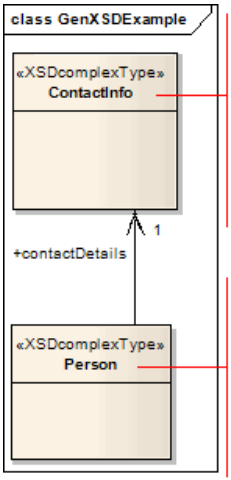
Enterprise Architect, by default, generates XML Schema in the Garden of Eden style. For every global XSDcomplexType stereotyped Class, the system generates a global element.

Example

You can change the specified default behavior by de-selecting the 'Generate global element for all global ComplexTypes' checkbox on the 'Generate XML Schema' dialog. Then, the generated XSD no longer contains the global element; that is, it no longer has the lines:

- <xs:element name="ContactInfo" type="ContactInfo"/> and
- <xs:element name="Person" type="Person"/>






```
<xs:complexType name="ContactInfo">
        <xs:sequence>
                <xs:element name="email" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="homePhone" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="mobilePhone" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="officePhone" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="streetAddress" type="xs:string" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
</xs:complexType>

<xs:complexType name="Person">
        <xs:sequence>
                <xs:element name="birthDate" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="firstName" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="gender" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="surName" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="contactDetails" type="ContactInfo" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
</xs:complexType>
```

Import XSD

To reverse engineer a W3C XML Schema (XSD) file to create or overwrite a Package of your UML Class model, you can use the XML Schema Import facility.

Access

Ribbon	Develop > Schema Modeling > Import XSD

Import Schema files

Option	Action
Package	Displays the name of the selected target Package.
Directory	Type in or browse for (click on) the directory containing the source XSD file(s).
Selected File(s)	Lists the XML Schema(s) currently available for import. • To select a single file, click on it

	• To select several individual files Ctrl+click on each file • To select a range of files, press Shift and select the first and last file in the range
Import global elements with "Type" postfix	Select this checkbox to treat the global element and the ComplexType it is referring to as two separate entities.
Import referenced XML Schema(s)	Select this checkbox to import any XML Schema that is being referenced by any of the files selected in the 'Selected File(s)' field.
Create Diagram for XML Schema(s)	Select this checkbox to create a Class diagram under each imported XSDschema Package.
Import XSD Elements/Attributes as	Select the appropriate radio button to indicate how the inline XSDelements and XSDattributes are to be imported into a Class, either as: • UML Associations or • UML attributes
Import	Click on this button to begin the XSD

	import.
Close	Click on this button to close the dialog, without saving your option selections.
Progress	Displays system messages indicating the progress of the Schema import. On imports containing a large number of external references, it can be useful to capture the progress messages to check exactly what has been imported. To do this, right-click on the messages and: • Copy the selected messages to the clipboard (select the 'Copy Selected to Clipboard' menu option) • Copy all the messages to the clipboard (select the 'Copy All to Clipboard' menu option), or • Save all the messages to a file (select the 'Save to File' menu option)

Notes

- If an XML Schema file being imported already exists in the model, Enterprise Architect skips importing the file
- References to XSD Primitive Types are always imported as UML attributes

- References to XSD constructs in external Schema files are always imported as UML attributes
- Enterprise Architect uses the schemaLocation attribute in the XSD Import and XSD Include elements of an XML Schema to determine the dependencies between the files; this attribute must be set to a valid file path (and not a URL) for the dependent XML Schema(s) to be imported correctly

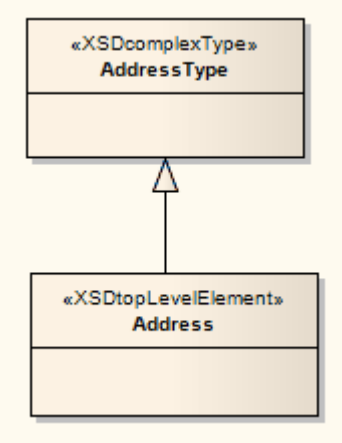
Global Element and ComplexType

Some XML Schemas have ComplexType elements with the same name as the referring global elements, but with the suffix 'Type', as shown:

```
<xs:element name="Address" type="AddressType"/>
<xs:complexType name="AddressType">
    <xs:sequence/>
</xs:complexType>
```

On XSD import, by default, Enterprise Architect treats this global element and its bounding ComplexType as a single entity, and creates a single XSDcomplexType stereotyped Class with the same name as the global element, as shown:

You can change this default behavior by selecting the 'Import global elements with "Type" postfix' checkbox on the 'Import XML Schema' dialog. When you select this option, the system treats the global element and the ComplexType it is referring to as two separate entities. For the example, the system creates an ?XSDtopLevelElement? stereotyped Class for the global element and an ?XSDcomplexType? stereotyped Class for the ComplexType, connected as shown:



Notes

- Enterprise Architect treats these two definitions as separate entities irrespective of whether the 'Import global elements with "Type" postfix' checkbox is selected or unselected:

```
    <xs:element name="HomeAddress"
type="AddressType"/>
    <xs:complexType name="AddressType">
        <xs:sequence/>
    </xs:complexType>
```

XSL Transforms

Model, Author and Execute XSLT Transforms and Stylesheets with XML Documents

Enterprise Architect provides facilities for modeling and executing XSL Transformations. XSLT is a technology that can be used to convert XML input documents into other types of document. Stylesheets are the XSL components used to transform the content. Facilities include:

- Specialized diagram and toolbox for modeling XSLT transformations
- Specialized editor for Stylesheet authoring, debugging and execution.
- XML document validation
- XML Schema validation

You model a transformation using the XML Transform diagram. On this diagram you can create xml documents and stylesheets, link them to a transformation (Activity) and then execute or debug the transformation. Inputs to the XSL Transform model are the XSLT and XML File Artifacts, which can be selected from the toolbox. These artifacts are most commonly created by dragging appropriate xml and xsl files on to the diagram. Output from the Transformation is described using the Output Artifact. The progress and success/failure of the transformation is shown on the 'XSLT' tab of the System Output window.

Create the XML Transform Diagram

Step	Action
1	In the Browser window, right-click on the appropriate Package and select the 'New Diagram' option.

Artifact Elements in the XML Transformation Toolbox

Artifact	Description
XML Transform	The model reference for the transformation, providing inputs and optional outputs. Used to run or debug the transformation. Inputs: XML File, XSLT Outputs: Output Artifact (Optional)
2	In the 'New Diagram' dialog, type an appropriate diagram name in the 'Name' field (if necessary) and click on 'Extended' in the 'Select From' list and

	'XML Transform' in the 'Diagram Types' list. Click on the OK button. The new diagram opens, with the Diagram Toolbox showing the 'XML Transformation' page.
XSLT	Identifies the stylesheet to execute. Inputs: N/A Outputs: N/A
XML File	Identifies the input document to transform. Inputs: N/A Outputs: N/A
XSD	Identifies the schema that can be used, optionally, to automatically perform XML validation on the output document. Inputs: Output Artifact, XML File, (or optional both) Outputs: N/A
Output Artifact	Use this Artifact to define the output of an XSLT operation. The Artifact provides the file path to use when output is created by the transform. To select or name the output file, double-click on the Artifact to display its properties and enter the file

	path under the 'Files' tab. To make use of the Artifact, draw a trace connector to it from the transform element.

Manually Validate documents

Using Enterprise Architect, you can perform XML validation both of documents to be transformed and of XSLT stylesheets.

To run the validation, right-click in the XML document or stylesheet in the XSL Debugger and select 'XML Validation'. A prompt displays to confirm if you are validating against a document type definition or an XML Schema.

- For a document type definition, simply click on the OK button; the validation proceeds
- For an XML schema, select the appropriate radio button to identify if the validation grammar is defined within the document or elsewhere; if elsewhere, enter the namespace and URL or file path for the grammar

If errors are found during a debugging run, they will be output to the Debug window (press Alt+8 to display this window).

If errors are found during a normal validation run, they will be output to the 'XSL' tab of the System Output window (press Alt+1 and select 'System Output' if this window does not display automatically). To locate the error in the

document, double-click on the error message.

XSLT Processor and Version

The XSL Processor used in these features is built from the Apache Xalan Project (C++ version 1.11)

Model an XSL Transformation

When you model an XSL Transformation, you can either draw on files that already exist in your file system, or you can create the contents of the stylesheet and source within the model elements.

Model elements from existing files

This is the simplest and most common method for modeling transformations. When you drag a file on to the XML Transform diagram, the appropriate Artifact element is generated for you. You can then use the Quick Linker to link the file Artifact elements as inputs to the XML Transform element, using Trace connectors.

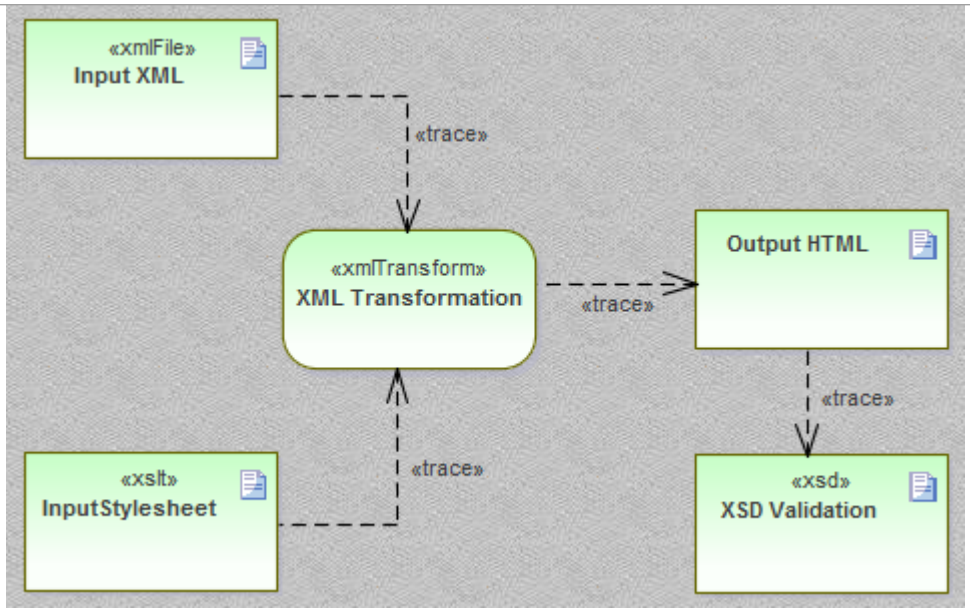
Optionally, you can:

- Specify an alternative output location (file) by linking an XML File or Output Artifact to a Trace connector from the XML Transform Artifact
- Validate the output document by dragging an XSD schema file on to the diagram and connecting the resulting XSD element to any Output Artifact of the XML Transform element

Step	Action
1	Open your file browser and the XML Transform diagram.

2	In the file browser, click on the input file and drag it onto the XML Transform diagram. A prompt displays to save the file as an: • External Artifact, where the XML File Artifact serves as a shortcut to the file in the file system • Internal Artifact, where the file content is read into the XML File Artifact and stored inside the model; you would select this option to make the source file contents available to other users of the model
3	Select the 'External Artifact' option. An XML File Artifact element is generated for the input file.
4	In the file browser, click on the XSL stylesheet file and drag it onto the XML Transform diagram. In response to the prompt, select the 'External Artifact' option. An XSLT element is generated for the stylesheet file.
5	Drag the XML Transform icon from the Toolbox onto the diagram, to create an XML Transform Activity element. If you prefer, give this element a new name.
6	Dragging the Trace icon from the 'Common'

	Toolbox page, create relationships between the: • Input XML file element and the transformation Activity element • XSLT Stylesheet file element and the transformation Activity element
7	(Optional.) • If you want to capture the output in a file, locate the appropriate file in the file browser and drag it onto the diagram to generate another File Artifact; link it to the XML Transform element with a Trace connector • If you want to validate the output document, locate the XSD schema file in the file browser and drag it on to the diagram to generate an XSD element; link this to the output File Artifact (or any Output Artifact) of the Transform element
8	Press Ctrl+S to save the diagram contents. If the output is intended to be HTML, your diagram might resemble this:



Modeling elements from scratch

When you use the 'XML Transformation' Toolbox to create XSLT and XML File elements, the system stores these as model documents. You double-click on the elements on the XML Transform diagram to open the model documents in Enterprise Architect's XSLT Debugger, where you can write and edit the file contents. When the document is saved, the contents will be saved back to the model.

Otherwise, the process of modeling a transformation is the same as described in *Modeling elements from existing files*.

Edit Documents with the XML Editor

Enterprise Architect provides a robust and useful XML editor with many features including:

- Intelli-sense
- Contextual structure tree providing quick alternative navigation (tip: press Ctrl+1 to toggle document tree view)
- Custom icons for XSL and XSD document elements
- Code completion and
- Validation of document and referenced schemas

The XML Editor will open when any document with an XML declaration is opened within Enterprise Architect. (Alternatively, press Ctrl+Shift+O.) The XSLT Debugger uses two XML editors side-by-side, to display both the stylesheet and the document being transformed.

Execute an XSL Transformation

After you have modeled an XSL Transformation, you can execute it directly from the model diagram. You can also perform the transformation directly from the XSL Stylesheet and input files.

Execute the Transformation From the Diagram

Step	Action
1	On the XML Transform diagram, right-click on the XML Transform Activity element and select the 'Run XSL Transformation' option. The XSLT Debugger view displays, showing the stylesheet (.xsl) file and XML document used in the transformation. The System Output window also displays, showing the error or success messages in the 'XSL' tab. (Press Ctrl+Shift+8 if the System Output window does not display.) If you have set up validation of the output, the System Output window also shows the validation comments.
	If you have directed the output to a file via an Output

2	or File Artifact, press F12 to view the output.

Debug an XSL Transformation

When you use the XSLT debugger to run a transformation you can control the process and inspect the state of the transformation using Enterprise Architect's debugger in combination with breakpoints. The XSLT Debugger provides a Run button and various Step buttons. You set breakpoints by clicking in the left margin of the stylesheet.

When a step completes or a breakpoint is encountered, the context of the transformation - including any parameters to template calls - can be viewed in the Locals window ('Execute > Windows > Local Variables'). You can also display the Call Stack ('Execute > Windows > Call Stack) to see how the current state of the transformation was reached.

Debug the Transformation

Step	Action
1	On the XML Transform diagram, right-click on the XML Transform Activity element and select the 'Debug XSL Transformation' option. The XSLT Debugger view displays, showing the stylesheet (.xsl) file and XML document used in the transformation, which is automatically initiated. The currently executing statement in the stylesheet is highlighted.

	Across the top of the view is a debugger toolbar, providing the normal debugging options to Start, Pause, Step Over, Step In, Step Out and Stop the debugging process. The final icon in the toolbar provides the option of hiding or showing the '.xml source' tab in the view. You can use these buttons to repeat and control the debugging process. The System Output window also displays, showing the debugger progress messages on the 'XSLT Transformation' tab. (Press Alt+1 and select 'System Output' if the System Output window does not display.) Error messages are directed to the Debug window (press Alt+8). You can also use the Debug window toolbar buttons and options to control debugging of the XSL Transformation.
2	If necessary, select to display the Locals window and the Call Stack. Click on the left margin of the XSLT Debugger stylesheet panel and set any Breakpoints you want to use to check processing.
3	Run the debugger again and examine the execution as indicated by the System Output window, Call Stack, Locals window, and any other Debugger or Execution Analysis tools you want to apply.

XML Validation

Enterprise Architect provides validation of XML documents. Documents can be verified against XML schema or Data Type Definitions (DTD). Validation is performed from within an Enterprise Architect editor using its context menu. Often an XML document will contain information relating to the schema that it conforms to. You can, however, choose to override this, validating the document against any schema, either at a path on your local machine or at a URL. This example demonstrates the use of the feature for a document that contains an incorrect attribute.

Access

Context Menu	Accessible from context menu of any editor window displaying xml content. Right-click in editor window and choose 'XML Validation'

XML Document Validation

Step	Action

1	Open the XML document to be validated.
2	Use the editor context menu and select the 'XML Validation' option.
3	Select the grammar of choice from the available options: • XML Schema (default) • Data Type Definition
4	Select the schema location. 'Defined in document' is selected by default. It is usual for an XML document to specify the schemas that govern its content. To choose a different schema from that defined in the document, select 'External' and provide either a URL or a file path. Examples: • http://mydomain/myschema.xsd • c:\mydomain\myschema.xsd
5	Click OK. The output of the validation will be displayed in the 'XML Validation' tab of the System Output window.

XML Document Validation Example

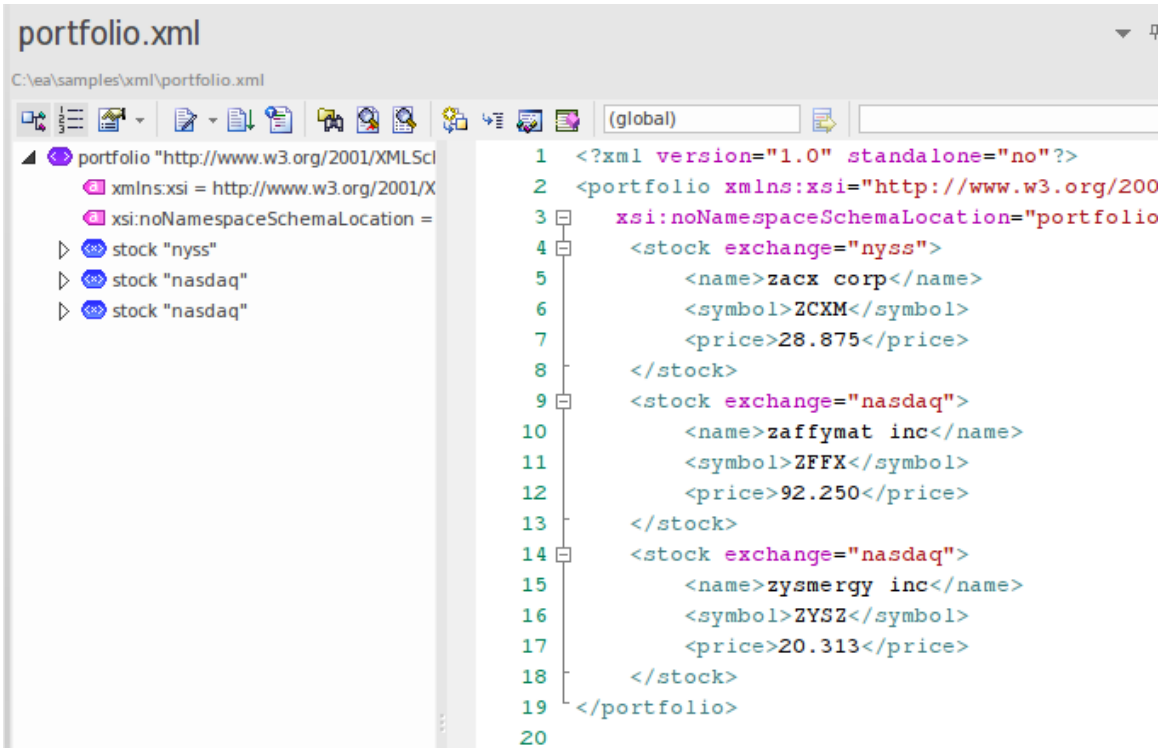


Figure 1: The XML document with an invalid attribute value 'nyss'

In this example, the document describes a stock item that has an invalid exchange code 'nyss'. As can be seen from this schema, the only valid values for the 'exchange' attribute are 'nyse', 'nasdaq' or 'ftsi'.

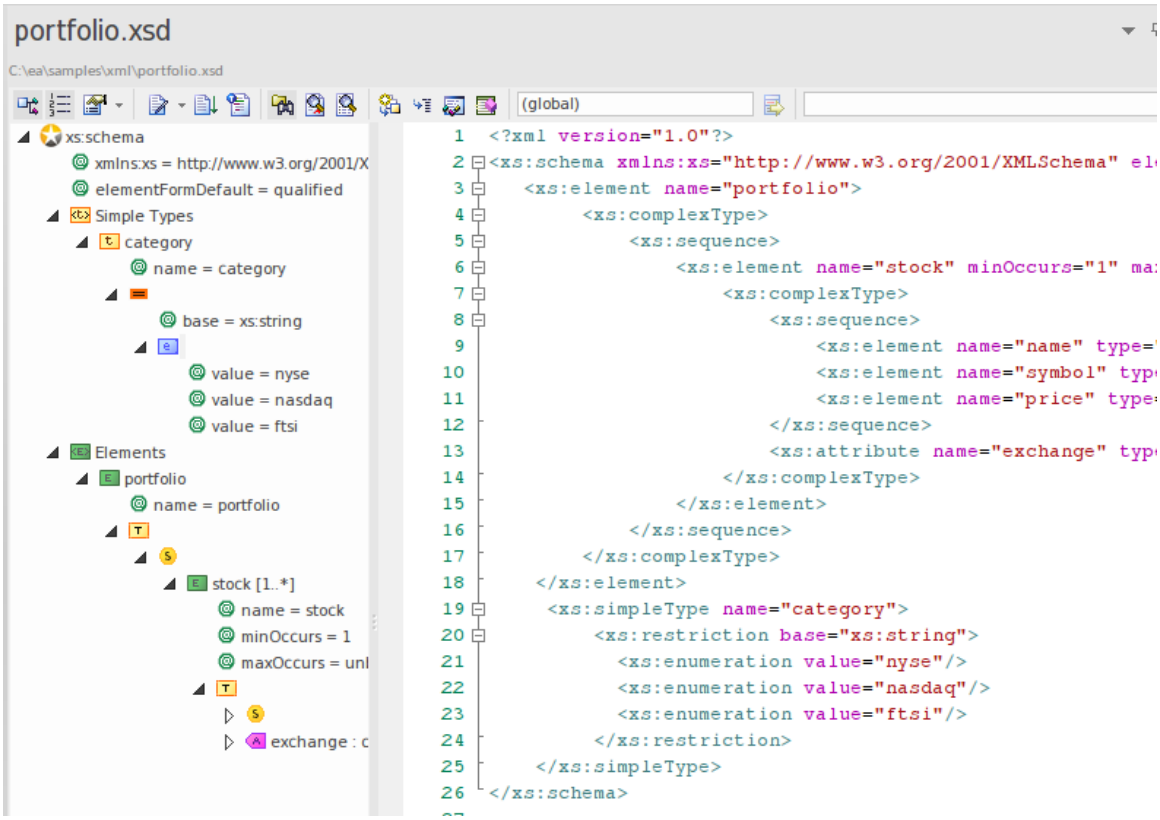


Figure 2: The XML Schema describing permitted stock exchange codes

This image shows the schema used in the validation. The declaration of a 'portfolio' element can be seen here to be made up of one or more 'stock' elements. Each stock element in turn, requires an 'exchange' attribute naming a code for the stock exchange in question.

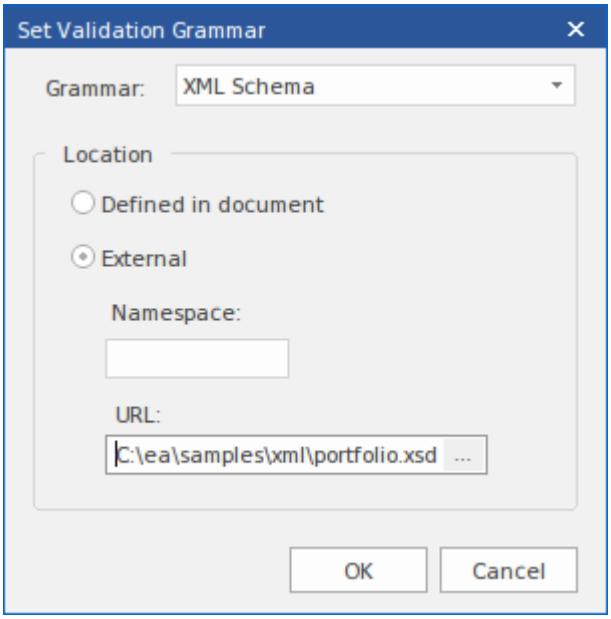


Figure 3: The 'XML validation' dialog naming a local schema file

This is the 'XML Validation' dialog. It is accessible from the context menu of any editor in Enterprise Architect that holds XML content. Here you can select the schema to use in the validation. In the example the processor will validate the document using a local schema file. This just happens to be the same schema named by the document, but it could be any schema (a development or later version of the schema for example).

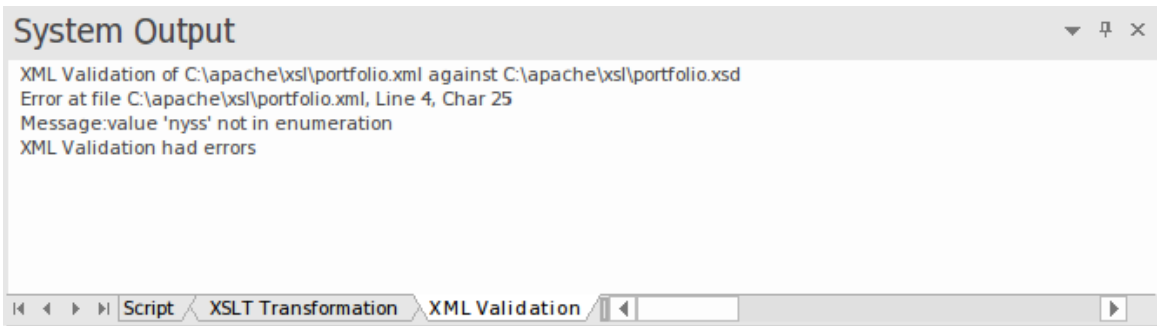


Figure 4: The System Output window showing validation error

This image shows the results of the validation. The attribute value 'nyss' has been identified as being incorrect according to the enumeration described by the schema.
Double-clicking the error will display the line of code in the editor where it can easily be corrected.

XML Service Oriented Architecture

Model Organizations, Communities and Systems to Maximize Agility, Scale and Interoperability

Service Oriented Architecture (SOA) is an architectural paradigm for defining how people, organizations and systems provide and use services to achieve results.

A service is an offer of value to another through a well-defined interface, available to a community (which could be the general public). A service results in work provided to one by another.

Service Oriented Architecture (SOA) is a way of organizing and understanding (representations of) organizations, communities and systems to maximize agility, scale and interoperability. The SOA approach is simple - people, organizations and systems provide services to each other. These services allow us to get something done without doing it ourselves or even without knowing how to do it - enabling us to be more efficient and agile. Services also enable us to offer our capabilities to others in exchange for some value - thus establishing a community, process or marketplace. The SOA paradigm works equally well for integrating existing capabilities as for creating and integrating new capabilities.

(Derived from *Service oriented architecture Modeling Language (SoaML) - Specification for the UML Profile and Metamodel for Services (UPMS)* (OMG document

ad/2008-11-01); pp. 25-26.)

In modeling and developing a complete Service Oriented Architecture in Enterprise Architect, you can work with any or all of:

- XML Schema Definition (XSD), also known as XML Schema - an XML technology that is used to specify the rules to which an XML document must adhere; Enterprise Architect provides a Schema Composer interface to help you model and generate XML schema
- XSL Transformations to convert input documents into XML or other types of document using XSL stylesheets, for which you use the XSLT Editor and Debugger for modeling and executing the transformations
- Web Services Description Language 1.1 (WSDL) - a key XML-based language for describing web services
- Service oriented architecture Modeling Language (SoaML) - a standard method of designing and modeling SOA solutions using the Unified Modeling Language (UML)
- Service-Oriented Modeling Framework (SOMF) - a service-oriented development life cycle methodology, offering a number of modeling practices and disciplines that contribute to successful service-oriented life cycle management and modeling
- National Information Exchange Modeling (NIEM) - a common framework that is used to define how information can be shared between systems, government agencies and departments

- Meta-Object Facility (MOF) - an Object Management Group (OMG) standard developed as a meta-modeling architecture to define the UML, and so provide a means to define the structure or abstract syntax of a language or of data

WSDL

Web Services Description Language 1.1 (WSDL) is a key XML-based, World Wide Web Consortium (W3C) language for describing web services. WSDL support is critical for the development of a complete Service Oriented Architecture (SOA), and the coupling of UML 2.5 and XML provides the natural mechanism for specifying, constructing and deploying XML-based SOA artifacts within an organization.

Using Enterprise Architect, you can rapidly model, forward engineer and reverse engineer WSDL files.

WSDL 1.1 Model Structure

A Web Service Description Language (WSDL), under specification 1.1, is defined within a ?WSDLnamespace? stereotyped Package, which represents the top-level container for the WSDL elements. Conceptually it maps to the targetNamespace in a WSDL definition element.

When you create a WSDL model, Enterprise Architect creates the Namespace and provides a set of sub-Packages, each containing a diagram on which to define the constituent elements of the model, with an Overview diagram to navigate between the sub-Packages. You work through the sub-Packages in sequence, to define the objects that are used by later objects, themselves called into still later objects.

WSDL Structure Development

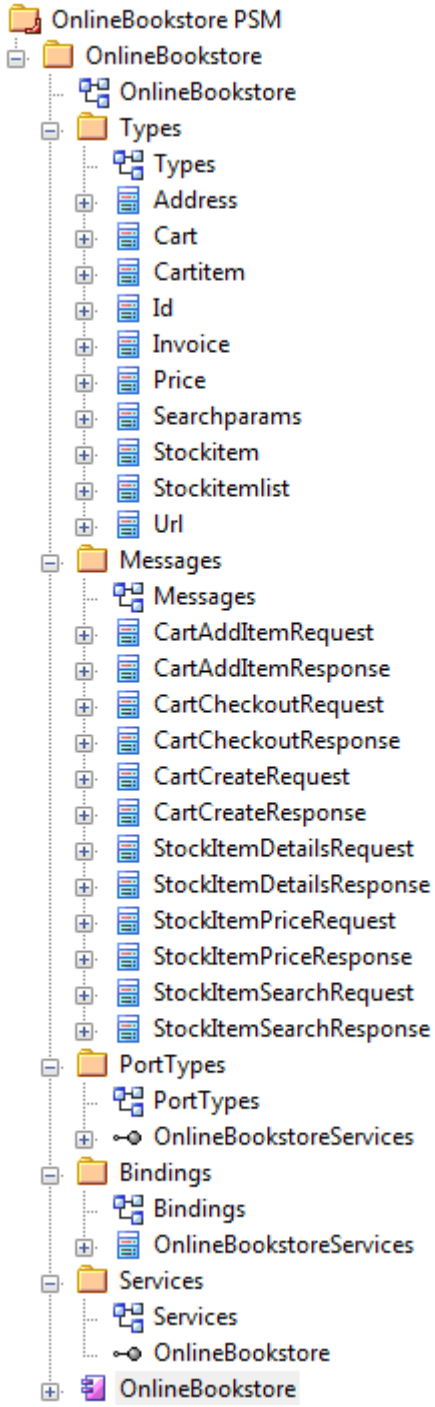
WSDL Element Type	Description
Types	Defined in an XSD Schema, these are the XSD data types used by the web service and communicated by WSDL Messages; you drag «XSDelement», «XSDsimpleType» and «XSDcomplexType» stereotyped elements onto the Types diagram from

	the 'XML Schema' page of the Diagram Toolbox.
Messages	WSDL Messages identify the data being communicated by a web service. Each Message element contains one or more Message Parts, which are attributes that each identify an XSD data type being communicated.
Port Types	WSDL Port Types are the essential core of the web service, defining the interfaces of the service. Each Port Type consists of a set of Port Type Operations, each of which identifies an exchange of Messages (data input to and output from the interface as that operation). The Port Type Operation can also identify Messages acting as Fault indicators.
Bindings	A Binding specifies the protocol and data format for the operations and messages defined for of a particular Port Type. Each «WSDLbinding » Class implements (realizes) the operations specified by the «WSDLportType» Interface - the Port Type Operations in the Port Type element are automatically copied into the Binding element as Binding operations.

Services	A WSDL Service defines a formal interface of the web service. It describes the collection of Port Types that expose a particular Binding, having an Association to each exposed Binding. It therefore encapsulates a set of the other data structures - if not all the data structures - defined in the model.
Documents	WSDL Documents are represented by Components having the stereotype «WSDL». This is the element from which you generate the WSDL file. You can create more than one Document to re-use the schema Types, Messages, Port Types, Bindings and Services of a Namespace across multiple physical WSDL documents, either in the same configuration or in different configurations.

Example

This figure shows an example WSDL namespace, OnlineBookstore PSM, which includes a single WSDL document, OnlineBookstore (at the bottom of the hierarchy).



Notes

- You can also generate a WSDL Package structure from a UML Interface using the WSDL Model Transformation

Model WSDL

You can quickly and easily model the elements in a Web Service Definition using the WSDL page of the Diagram Toolbox. As a first step, you can create an example WSDL Package structure in the Browser window, using the Namespace icon from the WSDL page. You can use this example Package structure as a template for developing your WSDL.

Create a new WSDL Package structure

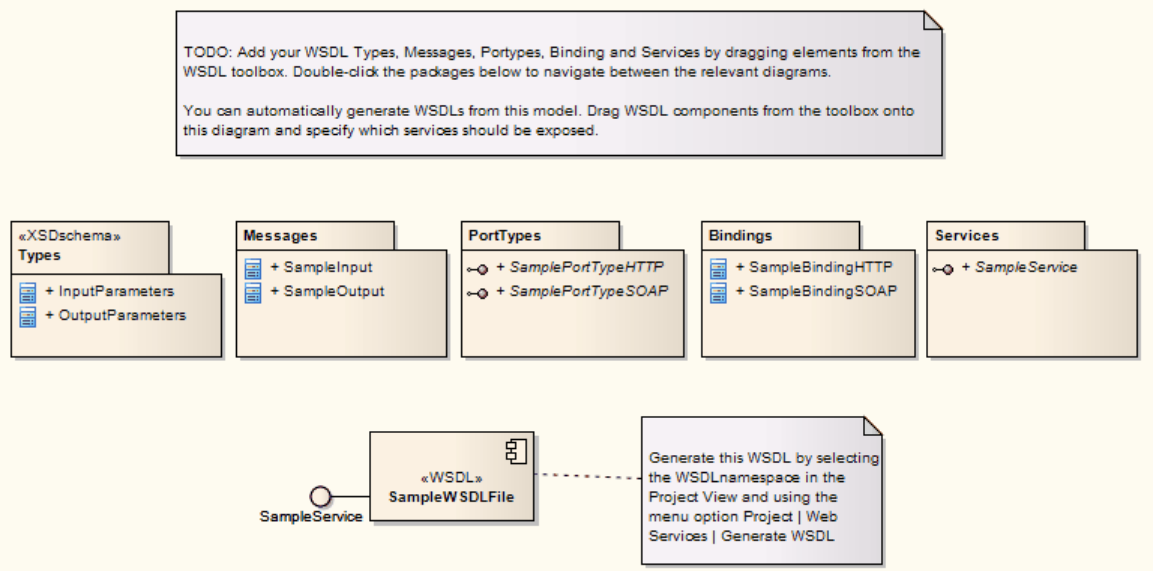
Step	Action
1	In the Browser window, create the top-level project structure you need (Model and Views), and click on the appropriate View.
2	Click on the 'New Package' option in the Browser window header drop-down list. The 'New Model Package' dialog displays.
3	In the 'Name' field type the name of the new Package, and select the 'Create Diagram' radio button.
4	Click on the OK button.

	The 'New Diagram' dialog displays.
5	In the 'Name' field type the name of the new diagram. In the 'Select From' panel select 'UML Structural', and in the 'Diagram Types' panel select 'Class'.
6	Click on the OK button. In the Browser window, double-click on the icon next to the new diagram's name; the diagram opens in the Diagram View, with the Class pages displaying in the Diagram Toolbox.
7	In the Toolbox, click on to display the 'Find Toolbox Item' dialog and specify 'WSDL', then select the Toolbox page from the results. The 'WSDL' Toolbox page displays.
8	Click on the 'Namespace' icon from the Toolbox and drag it into the Class diagram. The 'WSDL Namespace Properties' dialog displays. Type in a WSDL Package name and the URL of the Target Namespace. You can edit these values later.
9	Click on the OK button. The sample «WSDLnamespace» stereotyped Package structure is created on the diagram, and the full model structure is displayed, expanded, in the

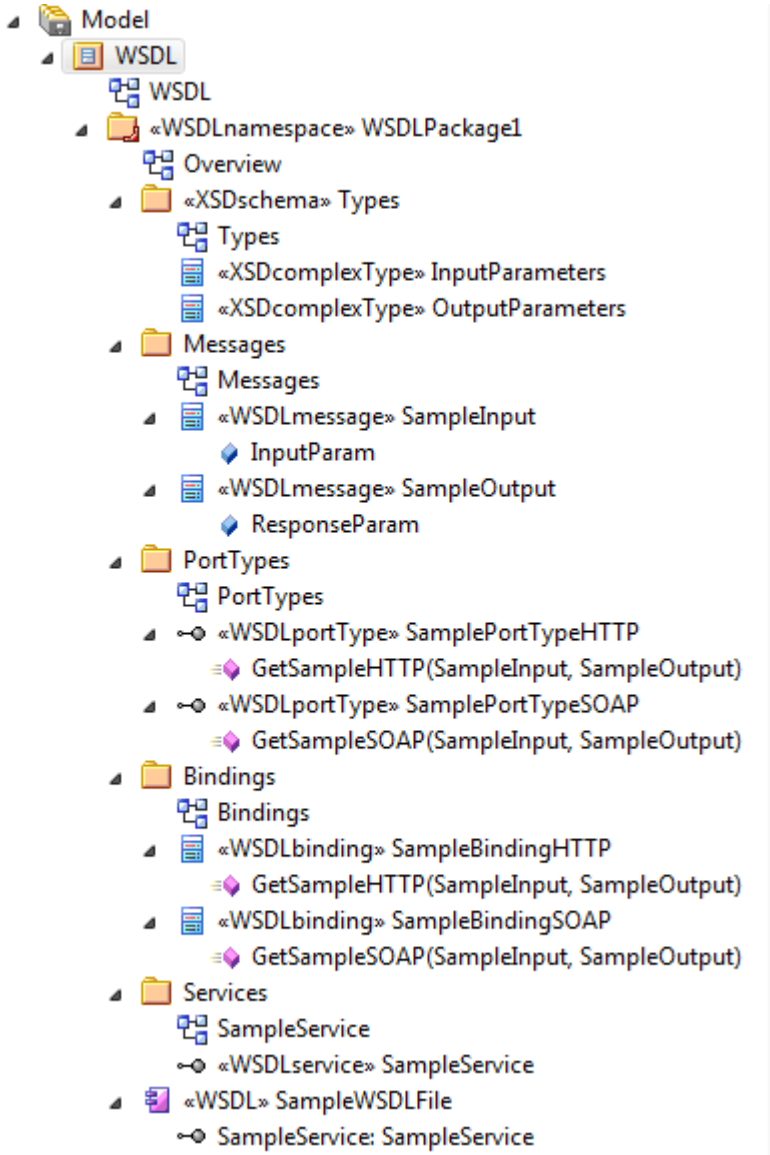
	Browser window. The model structure consists of these sub-Packages, with an Overview diagram to navigate between them: • Types: Contains the XSD types for the data communicated by the web service, on a Types diagram • Messages: Contains the WSDL Messages, modeled as UML Classes marked with the stereotype «WSDLmessage» • PortTypes: Contains the WSDL Port Types, modeled as UML interfaces marked with the stereotype «WSDLportType» • Bindings: Contains the WSDL Bindings, modeled as UML Classes that realize the PortTypes • Services: Contains the WSDL Services, modeled as UML interfaces with Associations to each exposed Binding
10	Model each of the WSDL constructs in their corresponding Packages.

Template WSDL Model - Diagram

The WSDLnamespace Package acts as a container for the WSDL structure.



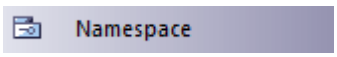
Template WSDL Model - Browser Window Hierarchy


Model
WSDL
WSDL
«WSDLnamespace» WSDLPackage1
Overview
«XSDschema» Types
Types
«XSDcomplexType» InputParameters
«XSDcomplexType» OutputParameters
Messages
Messages
«WSDLmessage» SampleInput
InputParam
«WSDLmessage» SampleOutput
ResponseParam
PortTypes
PortTypes
«WSDLportType» SamplePortTypeHTTP
GetSampleHTTP(SampleInput, SampleOutput)
«WSDLportType» SamplePortTypeSOAP
GetSampleSOAP(SampleInput, SampleOutput)
Bindings
Bindings
«WSDLbinding» SampleBindingHTTP
GetSampleHTTP(SampleInput, SampleOutput)
«WSDLbinding» SampleBindingSOAP
GetSampleSOAP(SampleInput, SampleOutput)
Services
SampleService
«WSDLservice» SampleService
«WSDL» SampleWSDLFile
SampleService: SampleService

WSDL Namespace

A «WSDLnamespace» stereotyped Package represents the top-level container for the WSDL constructs in Enterprise Architect. You can create the Namespace Package by dragging the Namespace icon from the WSDL Toolbox page and dropping it directly onto a diagram.

Toolbox Icon



Access

To display the 'WSDL Namepace Properties' dialog for the selected «WSDLnamespace» stereotyped Package, use one of the methods outlined here.

Ribbon	Design > Package > Manage > Properties
Context Menu	Right-click on «WSDLnamespace» stereotyped Package | Properties
Other	In Browser window, double-click on «WSDLnamespace» stereotyped Package, or Drag Namespace icon from toolbox onto a diagram (this creates a new «WSDLnamespace» stereotyped

	Package)

Define Properties

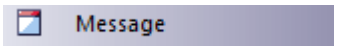
Option	Action
WSDL Package Name	Type in the name of the WSDL Namespace Package element.
Target Namespace	(Optional) Type in the URL for the WSDL Namespace Package.
OK	Click on this button to save the values entered and close the WSDL Namespace 'Properties' dialog. If you have just created the Namespace, a new Package diagram opens containing the sample template WSDL model.
Cancel	Click on this button to discard the data entered and close the 'WSDL Namespace Properties' dialog.
Help	Click on this button to display this Help topic.

UML	This button is displayed when you are editing existing WSDL Namespace element information. Click on the button to open the UML element 'Properties' dialog for the Namespace Package element.

WSDL Message

A «WSDLmessage» stereotyped Class represents a WSDL Message and acts as a container for one or more WSDL Message Parts. You can create WSDL Messages by dragging the Message icon from the WSDL Toolbox and dropping it directly onto the Messages diagram (under the Messages Package in the WSDL Package structure).

Toolbox Icon



Access

To display the 'WSDL Message' dialog for the selected «WSDLmessage» stereotyped Class, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties
Context Menu	Right-click on «WSDLmessage» stereotyped Class| Properties
Keyboard Shortcuts	Alt+Enter
Other	Double-click on a ?WSDLmessage? stereotyped Class, or

	Drag icon from the toolbox and drop directly onto the Messages diagram, under the Messages Package in the WSDL Package structure. (This creates a new ?WSDLmessage? stereotyped Class.)

Define Properties

Option	Action
Name	Type in the name of the WSDL Message.
Documentation	(Optional) Type in any notes you need for this element.
OK	Click on this button to save the data entered and close the WSDL Message dialog.
Cancel	Click on this button to discard the data entered and close the 'WSDL Message' dialog.
Help	Click on this button to display this Help topic.

UML	This button is displayed when you are editing existing WSDL Message element information. Click on the button to open the UML Class 'Properties' dialog for the element.

Notes

- WSDL Messages can only be created under the Messages Package in the WSDL Package structure
- The name of the WSDL Message should be unique amongst all WSDL Messages within the WSDL

WSDL Message Part

A WSDL Message Part is the segment of a WSDL Message that identifies the XSD data type of the data communicated by the Message. If a Message communicates data of more than one data type, each data type is identified in a separate Message Part.

In Enterprise Architect, a WSDL Message Part is represented by a UML attribute of the WSDL Message Class. You can create the Message Part attribute by dragging the 'Message Part' icon from the WSDL Toolbox and dropping it onto a ?WSDLmessage? stereotyped Class.

Toolbox Icon

Access

To display the 'WSDL Message Part' dialog for the selected Message Part, use one of the methods outlined here.

Ribbon	With a specific Message Part (attribute) selected within a WSDL Message on a diagram: Design > Element > Features > Attributes
Context	With a specific Message Part (attribute) selected within a WSDL Message on a

Menu	diagram: Right-click on attribute | View Properties
Keyboard Shortcuts	With a specific Message Part (attribute) selected within a WSDL Message on a diagram: F9
Other	Double-click on the Message Part (attribute) within the ?WSDLmessage? stereotyped Class, or Drag icon from toolbox and drop onto a ?WSDLmessage? stereotyped Class (This creates a new Message Part (as an attribute) within the ?WSDLmessage? stereotyped Class.)

Define Properties

Option	Action
Name	Type in the name of the WSDL Message Part attribute.
Type	Either: • Type the name of a data type, or

	• Click on the drop-down arrow and select an XSD built-in dataType from the list, or • Click on the ... button and browse for an existing ?XSDelement?, ?XSDcomplexType? or ?XSDsimpleType? element as a classifier The XSD Types can be defined in: • The Types Package under the WSDL Package Structure or • Any other Package in the model
OK	Click on this button to save the data entered and close the 'WSDL Message Part' dialog.
Cancel	Click on this button to discard the data entered and close the 'WSDL Message Part' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing WSDL Message Part attribute information. Click on the button to open the attribute

	properties for the Message Part.

Notes

- WSDLmessage stereotyped Classes can effectively contain Message Part attributes only; if you add other attributes to the Class element, they are re-cast as Message Parts

WSDL Port Type

A «WSDLportType» stereotyped Interface represents a WSDL PortType. It describes the operations exposed by the WSDL, acting as a container for one or more WSDL Port Type Operations. You can create a WSDL PortType element by dragging the Port Type icon from the WSDL Toolbox and dropping it directly onto the PortTypes diagram (under the PortTypes Package in the WSDL Package structure).

Toolbox Icon

Access

To display the 'WSDL PortType' dialog for the selected «WSDLportType» stereotyped Interface, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties
Context Menu	Right-click on «WSDLportType» stereotyped Interface| Properties
Keyboard Shortcuts	Alt+Enter

Other	• Double-click on a ?WSDLportType? stereotyped Interface, or • Drag icon from the toolbox and drop directly onto the PortTypes diagram, under the PortTypes Package in the WSDL Package structure (This creates a new ?WSDLportType? stereotyped Interface.)

Define Properties

Option	Action
Name	Type in the name of the WSDL PortType.
Documentation	(Optional) Type in any notes you need for this element.
OK	Click on this button to save the data entered and close the WSDL PortType dialog.
Cancel	Click on this button to discard the data entered and close the 'WSDL PortType' dialog.

Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing WSDL PortType element information. Click on the button to open the UML element 'Properties' dialog for the element.

Notes

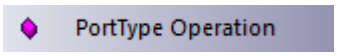
- WSDL PortTypes can only be created under the PortTypes Package in the WSDL Package structure
- The name of the WSDL PortType should be unique amongst all the WSDL PortTypes within the WSDL

WSDL Port Type Operation

A Port Type Operation identifies an exchange of Messages (data input to and output from the interface as an operation). The Port Type Operation can also identify Messages acting as Fault indicators.

In Enterprise Architect, a WSDL PortType Operation is represented by a UML Operation of the WSDL PortType Interface. You can create a PortType Operation by dragging the PortType Operation icon from the WSDL Toolbox and dropping it onto a ?WSDLportType? stereotyped Interface.

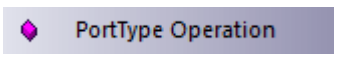
Toolbox Icon



Access

To display the 'WSDL PortType Operation' dialog for the selected PortType Operation, use one of the methods outlined here.

Ribbon	With a specific PortType Operation selected within a ?WSDLportType? stereotyped Interface on a diagram: Design > Element > Features > Operations
Context	With a specific PortType Operation

Menu	selected within a ?WSDLportType? stereotyped Interface on a diagram: Right-click on attribute | View Properties
Keyboard Shortcuts	With a specific PortType Operation selected within a «WSDLportType» stereotyped Interface on a diagram: F10
Other	Double-click on the PortType Operation within the ?WSDLportType? stereotyped Interface, or Drag  icon from toolbox and drop onto a ?WSDLportType? stereotyped Interface. (This creates a new PortType Operation (as a UML operation) within the ?WSDLportType? stereotyped Interface.)

Define Properties

Option	Action
Name	Type in the name of the WSDL PortType Operation.

Documentation	(Optional) Type in any notes you need for this operation.
Operation Type	Click on the drop-down arrow and select one of the supported PortType Operation types: • OneWay • Request-Response • Solicit-Response • Notification
Input	This section is grayed out if you have selected Notification as the operation type. • Name - Defaults to a name that parallels theOperation Type. If you do not want to use the default, type an alternative name for the input Message. • Message - Click on the drop-down arrow and select one of the WSDL Messages previously created in theMessagePackage. • Documentation - (Optional) Type in any notes you need for this input Message.
Output	This section is grayed out if you have selected OneWay as the operation type.

	• Name - Defaults to a name that parallels theOperation Type. If you do not want to use the default, type an alternative name for the output Message. • Message - Click on the drop-down arrow and select one of the WSDL Messages previously created in theMessagePackage. • Documentation - (Optional) Type in any notes you need for this output Message.
Faults	Review the details of the WSDL Messages that can act as Faults. Faults display in this list with the most recently-created at the top and the oldest at the end. If more than four Fault Messages are defined, use the vertical scroll bar to display the rest of the list. To add a Message, click on the New button. The 'WSDL PortType Operation Fault' dialog displays. • Name - Defaults to 'Fault<n>'; if you do not want to use the default, type an alternative name for the fault Message • Message - Click on the drop-down arrow and select one of the WSDL Messages previously created in the

	Message Package • Documentation - (Optional) Type in any notes you need for this fault Message • OK - Click on this button to save the data entered and close the 'WSDL PortType Operation Fault' dialog • Cancel - Click on this button to discard the data entered and close the 'WSDL PortType Operation Fault' dialog • Help - Click on this button to display this Help topic To remove a Message from the list, click on it and click on the Delete button.
OK	Click on this button to save the data entered and close the 'WSDL PortType Operation' dialog.
Cancel	Click on this button to discard the data entered and close the 'WSDL PortType Operation' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing WSDL Port Type

	Operation information. Click on the button to open the UML operation 'Properties' dialog for the element.

Notes

- WSDL PortType Operations can only be contained by WSDL PortTypes
- The name provided for an Input, Output or Fault Message in a PortType Operation must be unique amongst the Input, Output and Fault Messages, respectively, across the WSDL PortType
- In the UML operation 'Properties' dialog, the Messages identified as Input, Output and Fault can be examined as the parameters of the operation

WSDL Binding

A WSDL Binding element implements the operations specified by a particular ?WSDLportType? stereotyped Interface and describes the message format and protocol details for the operations and messages defined by this WSDL PortType. A WSDL Binding element is represented by a ?WSDLbinding? stereotyped Class.

You create a WSDL Binding element by dragging the Binding icon from the WSDL Toolbox directly onto the Bindings diagram under the Bindings Package in the WSDL Package structure.

Toolbox Icon

Access

To display the 'WSDL Binding' dialog for the selected «WSDLbinding» stereotyped Class, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties
Context Menu	Right-click on «WSDLbinding» stereotyped Class | Properties
Keyboard	Alt+Enter

Shortcuts	
Other	Double-click on a ?WSDLbinding? stereotyped Class, or Drag icon from the toolbox and drop directly onto the Bindings diagram, under the Bindings Package in the WSDL Package structure. (This creates a new ?WSDLbinding? stereotyped Class.)

Define Properties

Option	Action
Name	Type in the name of the WSDL Binding element.
PortType	Click on the drop-down arrow and select the WSDL PortType to be implemented by this WSDL Binding.
Protocol	Click on the drop-down arrow and select the protocol for the transmission of the selected WSDL PortType's operations. The supported protocols are:

	• SOAP • HTTP
Transport	This field is disabled if you have selected the HTTP protocol. Defaults to http://schemas.xmlsoap.org/soap/http. If necessary, type in an alternative URL for the SOAP protocol.
Style	This field is disabled if you have selected the HTTP protocol. Click on the drop-down arrow and select the style of SOAP protocol.
Verb	This field is disabled if you have selected the SOAP protocol. Click on the drop-down arrow and select the appropriate HTTP verb. The supported verbs are: • GET • POST
Documentation	(Optional) Type in any notes you need for this element.
OK	Click on this button to save the data entered and close the 'WSDL Binding'

	dialog.
Cancel	Click on this button to discard the data entered and close the 'WSDL Binding' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing WSDL Binding element information. Click on the button to open the UML element 'Properties' dialog for the element.

Notes

- A WSDL Binding must implement a WSDL PortType; therefore, WSDL PortTypes should be defined before you create WSDL Bindings
- WSDL Bindings can only be created under the Bindings Package in the WSDL Package structure
- The name of the WSDL Binding should be unique amongst all the WSDL Bindings within the WSDL

WSDL Binding Operation

When you save a newly-created ?WSDLbinding? stereotyped Class, the system:

1. Adds to the Binding diagram, the WSDL Port Type element implemented by the WSDL Binding.
2. Draws a Realization connector from the Binding to the PortType.
3. Automatically populates the Binding with all the UML operations from the PortType, as the WSDL Binding Operations.

Access

To display the 'WSDL Binding Operation Details' dialog for the selected Binding Operation, use one of the methods outlined here.

Ribbon	With a specific Binding Operation selected within a ?WSDLbinding? stereotyped Class on a diagram: Design > Element > Features > Operations
Context Menu	With a specific Binding Operation selected within a ?WSDLbinding? stereotyped Class on a diagram: Right-click on attribute | View Properties

Keyboard Shortcuts	With a specific Binding Operation selected within a ?WSDLbinding? stereotyped Class on a diagram: F10
Other	Double-click on the Binding Operation within the «WSDLbinding» stereotyped Class

Define Properties

Option	Action
Operation Name	Displays the name of the Operation copied from the WSDL PortType element. The value in this field cannot be edited.
Action	If the protocol of the parent WSDL Binding element was defined as HTTP, this field is grayed out. Type in the SOAP Action header (URL) for this operation.
Style	If the protocol of the parent WSDL Binding element was defined as HTTP,

	this field is grayed out. Click on the drop-down arrow and select the SOAP style of the operation.
Location	If the protocol of the parent WSDL Binding element was defined as SOAP, this field is grayed out. Type in the relative URL of this Operation.
Documentation	(Optional) Type in any notes you need for this operation.
Parameters	Click on this button to define the parameters for this operation. The 'WSDL Binding Operation Parameters' dialog displays, showing the names of the operation Input, Output and Faults. You cannot change these names. Click on the Details button to specify the details for Input, Output and Fault operation (Message) parameters. Note that the Details button in the: • Input section is disabled for the Notification Operation Type • Output section is disabled for the One-way Operation Type • Fault section is disabled if there are no

	Fault Messages • Use - If the protocol of the parent WSDL Binding element was defined as HTTP, this field is grayed out; click on the drop-down arrow and select the encoding that is to be used • Encoding Style - If the protocol of the parent WSDL Binding element was defined as HTTP, this field is grayed out; if 'Use' is set to 'encoded', type in the style (URL) to apply • Namespace - If the protocol of the parent WSDL Binding element was defined as HTTP, this field is grayed out; (Optional) type in the namespace • Parts - If the protocol of the parent WSDL Binding element was defined as HTTP, this field is grayed out; this field is also not applicable to Faults - (Optional) type in the Message Part attributes that appear within the SOAP Body portion • Header - This field is not applicable to Faults; (Optional) type in the text of the SOAP/HTTP Header • Documentation - (Optional) Type in any notes you need for this message

	• OK - Click on this button to save the data entered and close the 'WSDL Binding Parameter Details' dialog • Cancel - Click on this button to discard the data entered and close the 'WSDL Binding Parameter Details' dialog • Help - Click on this button to display this Help topic
OK	Click on this button to save the data entered and close the 'WSDL Binding Operation Details' dialog.
Cancel	Click on this button to discard the data entered and close the 'WSDL Binding Operation Details' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing WSDL Binding Operation information. Click on the button to open the UML operation 'Properties' dialog for the element.

Notes

- If you subsequently change the WSDL Port Type operations, you can refresh the Binding Operations by deleting the Realization connector and re-establishing it; the 'Overrides & Implementations' dialog displays, on which you select the updated operations to establish
- You can review the parameters of a Binding Operation by highlighting the operation in the diagram or Browser window and expanding the entries in the Properties window

WSDL Service

A WSDL Service is represented by a ?WSDLservice? stereotyped Interface; it describes a collection of Ports that expose a particular Binding. You can create a WSDL Service element by dragging the Service icon from the WSDL Toolbox and dropping it directly onto a diagram in the Services Package of your WSDL model.

When you save a newly-created ?WSDLservice? stereotyped Interface, the system:

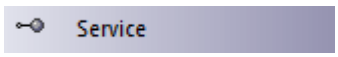
1. Adds the WSDL Binding elements exposed by the WSDL Service to the Service diagram.
2. Draws an Association connector from the Service element to each Binding element.
3. Labels each connector with the corresponding Port name.

Toolbox Icon

Access

To display the 'WSDL Service' dialog for the selected «WSDLservice» stereotyped Interface, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties

Context Menu	Right-click on «WSDLservice» stereotyped Interface| Properties
Keyboard Shortcuts	Alt+Enter
Other	Double-click on a ?WSDLservice? stereotyped Interface, or Drag the  icon from the toolbox and drop directly onto the SampleService diagram, under the Services Package in the WSDL Package structure. (This creates a new ?WSDLservice? stereotyped Interface.)

Define Properties

Option	Action
Name	Type in the name of the WSDL Service.
Documentation	(Optional) Type in any notes you need for this element.
Ports	Identify the Ports (or endpoints) for this WSDL Service.

	To add a Port to the list, click on the New button. The 'WSDL Port' dialog displays. • Port Name - Type in the name of the Port • Binding - Click on the drop-down arrow and select a Binding element from the list of all the WSDL Bindings created in theBindingsPackage • Location - Type in the URL for the Port • Documentation - (Optional) Type in any notes you need for this Port • OK - Click on this button to save the values entered and close the 'WSDL Port' dialog • Cancel - Click on this button to discard the values entered and close the 'WSDL Port' dialog • Help - Click on this button to display this Help topic The Ports are organized in the list with the most recent at the top and the oldest at the end. To remove an entry from the list, click on it and click on the Delete button.
OK	Click on this button to save the data entered and close the WSDL Service

	dialog.
Cancel	Click on this button to discard the data entered and close the 'WSDL Service' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing WSDL Service element information. Click on the button to open the UML element 'Properties' dialog for the element.

Notes

- WSDL Services can only be created under the Service Package in the WSDL Package structure
- The name of the WSDL Service should be unique amongst all the WSDL Services within the WSDL

WSDL Document

A WSDL Document encapsulates a Web Service defined within the ?WSDLnamespace? stereotyped Package, and is the source from which the WSDL file is generated. It is represented by a ?WSDL? stereotyped Component element as a direct child element of the ?WSDLnamespace? stereotyped Package. You can have multiple WSDL Documents under a single WSDL Namespace, to reuse and expose the WSDL Services for that namespace across multiple WSDLs.

One ?WSDL? stereotyped Component element is automatically created when you create the Namespace Package structure. You can add further WSDL elements by dragging the WSDL icon from the WSDL Toolbox and dropping it directly onto the namespace Overview diagram.

Toolbox Icon

Access

To display the 'WSDL Document Properties' dialog for the selected «WSDL» stereotyped Component, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties

Context Menu	Right-click on «WSDL» stereotyped Component | Properties
Keyboard Shortcuts	Alt+Enter
Other	Double-click on a ?WSDL? stereotyped Component, or Drag icon from the toolbox and drop directly onto the Overview diagram, under the ?WSDLnamespace? stereotyped Package in the WSDL Package structure. (This creates a new WSDL Document, represented by a ?WSDL? stereotyped Component.)

Define Properties

Option	Action
Name	Type in the name of the WSDL document.
File Name	Type the file path into which the WSDL 1.1 file is to be generated.

Documentation	(Optional) Type in any notes you need for this element.
XMLNS	Identify the additional namespace or namespace-prefix pairs used in this WSDL Document. To add a namespace or namespace-prefix pair, click on the New button; to edit an existing entry, double-click on it. In either case, the 'Namespace Details' dialog displays. • Prefix - Type in the abbreviated text to represent the Namespace • Namespace - Type in the name of the Namespace • OK - Click on this button to save the new information and close the 'Namespace Details' dialog • Cancel - Click on this button to discard the new information and close the 'Namespace Details' dialog • Help - Click on this button to display this Help topic To remove an entry from the list, click on it and click on the Delete button.
Services	Review the WSDL Services that exist in the Services Package.

	Select the checkbox against the services to be included in the current WSDL file.
OK	Click on this button to save the data entered and close the WSDL Document 'Properties' dialog.
Cancel	Click on this button to discard the data entered and close the 'WSDL Document Properties' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing WSDL Document element information. Click on the button to open the UML element 'Properties' dialog for the element.

Generate WSDL

If you have developed a WSDL model in UML, you can forward-engineer it into WSDL 1.1 files using the Generate WSDL feature. This feature acts on either a «WSDLnamespace» stereotyped Package or a «WSDL» stereotyped Component (Document), and generates any or all of the WSDL Components owned by the target «WSDLnamespace» structure.

Access

Ribbon	Develop > Schema Modeling > Export WSDL

Generate WSDL 1.1 files

Option	Action
WSDL Package	Displays the name of the WSDL Namespace containing the source Component(s) from which the WSDL file is to be generated.
Encoding	Either:

	• Click on the drop-down arrow and select the XML encoding scheme you need, or • Click on the Default button to apply the default encoding scheme (UTF-8)
Select Components To Generate	Click on the ?WSDL? stereotyped Component(s) in the list for which the WSDL file is to be generated. To: • Select multiple individual Components use Ctrl+click • Select a range use Shift+click • Select all entries in the list click on the Select All button • Clear all entries in the list click on the Select None button • Provide a file path and name into which to generate the WSDL file for a component, double-click on the component name; the 'Component File Name' dialog is displayed, see the table for a description
Generate	Click on this button to generate the WSDL files for the selected ?WSDL? stereotyped Components. A message displays when the generation

	is complete; click on the OK button on the message to close it.
View WSDL	Click on this button to display the most recently generated WSDL.
Close	Click on this button to close this dialog.
Help	Click on this button to display this Help topic.
Progress	Monitor the progress of the WSDL file generation.

Component File Name dialog

Field/Button	Description
Name	Displays the name of the selected «WSDL» stereotyped Component.
Prefix	If multiple prefixes have been specified, click on the drop-down arrow and select the appropriate prefix for the WSDL Namespace.
	Type in or browse for (click on ...) the

File Name	file path and name into which the WSDL file is to be generated.
OK	Click on this button to save the data entered and close the 'Component File Name' dialog.
Cancel	Click on this button to discard the data entered and close the 'Component File Name' dialog.
Help	Click on this button to display this Help topic.

Notes

- You can also generate WSDL files through the Automation Interface

Import WSDL

If you have WSDL 1.1 files external to Enterprise Architect that you want to reverse engineer into UML Class models, you can import them into the system using the WSDL Import facility.

Access

Browser window | Click on root Package to contain imported file, then:

Ribbon	Develop > Schema Modeling > Import WSDL

Import a WSDL File

Option	Action
Package	Displays the name of the root Package under which the WSDL file is to be imported.
Filename	Type in or browse for (click on ...) the name and path of the WSDL file to import.

Target Package	Defaults to the name of the WSDL file being imported, as the name of the Package to represent the imported file. If you do not want to use the default name, type in a different name.
Import	Click on this button to start the WSDL Import. A message displays when the import is complete; click on the OK button on the message to close it.
Close	Click on this button to close this dialog.
Progress	Monitor the progress of the WSDL Import.

Notes

- Enterprise Architect cannot import a WSDL file that references WSDL constructs existing outside that file; if there are referenced constructs in other files, combine all files into a single file and import that combined file
- Example of an importable file: http://www.w3.org/TR/wsdl.html#_wsdl
- Example of a non-importable file: http://www.w3.org/TR/wsdl.html#_style; attempts to

import this file result in the error message Cannot Import Split Files

SoaML

Service oriented architecture Modeling Language (SoaML) is a standard method of designing and modeling SOA solutions using the Unified Modeling Language (UML).

This text is derived from Service oriented architecture Modeling Language (SoaML) - Specification for the UML Profile and Metamodel for Services (UPMS) (OMG document ad/2008-11-01); pp. 25-26:

"A service is an offer of value to another through a well-defined interface and available to a community (which may be the general public). A service results in work provided to one by another."

"Service Oriented Architecture (SOA) is a way of organizing and understanding (representations of) organizations, communities and systems to maximize agility, scale and interoperability. The SOA approach is simple - people, organizations and systems provide services to each other. These services allow us to get something done without doing it ourselves or even without knowing how to do it - enabling us to be more efficient and agile. Services also enable us to offer our capabilities to others in exchange for some value - thus establishing a community, process or marketplace. The SOA paradigm works equally well for integrating existing capabilities as for creating and integrating new capabilities."

"SOA ... is an architectural paradigm for defining how people, organizations and systems provide and use services

to achieve results. SoaML ... provides a standard way to architect and model SOA solutions using the Unified Modeling Language (UML). The profile uses the built-in extension mechanisms of UML to define SOA concepts in terms of existing UML concepts."

"... the highest leverage of employing SOA comes from understanding a community, process or enterprise as a set of interrelated services and ... supporting that service oriented enterprise with service-enabled systems. SoaML enables business oriented and systems oriented services architectures to mutually and collaboratively support the enterprise mission. ... SoaML depends on Model Driven Architecture® (MDA®) to help map business and systems architectures, the design of the enterprise, to the technologies that support SOA, such as web services and CORBA®."

"For further information on the concepts of SoaML, see the specification document on the OMG website SoaML document page."

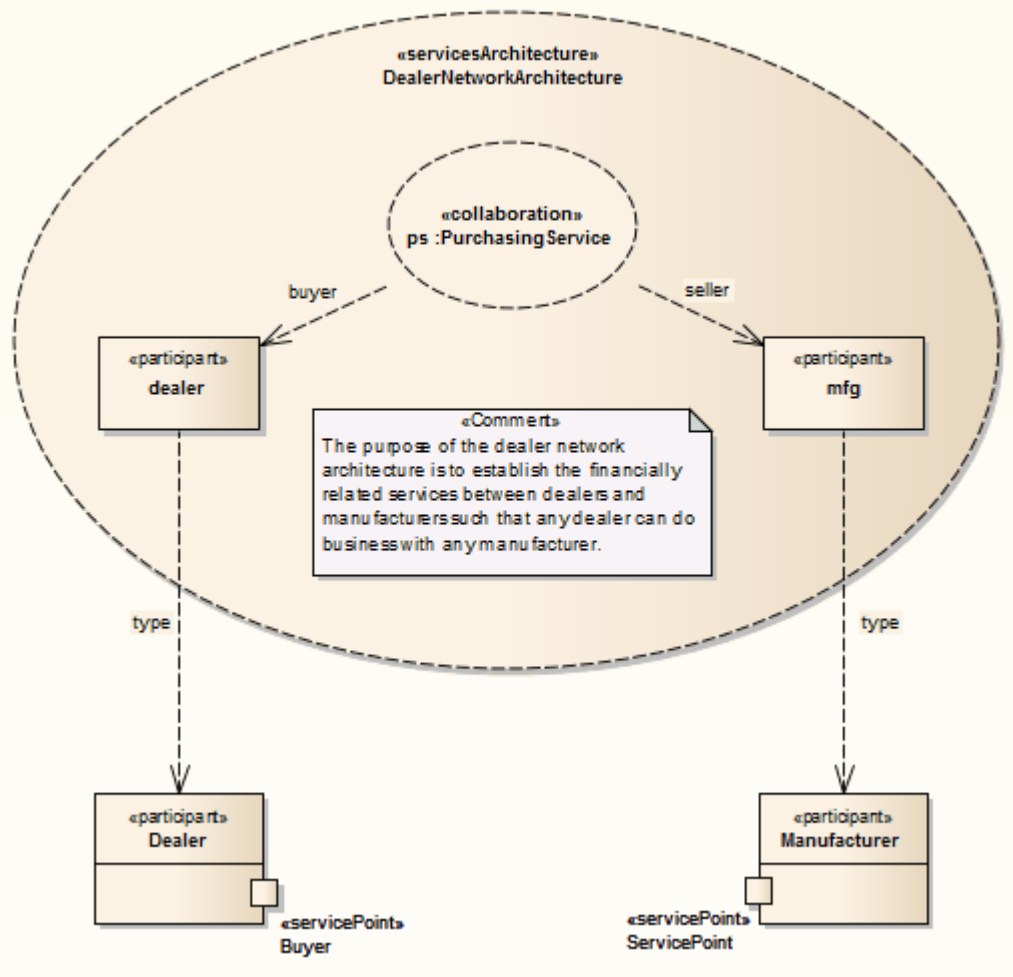
SoaML in Enterprise Architect

In Enterprise Architect you can model services architectures quickly and simply through use of an MDG Technology integrated with the Enterprise Architect installer. The SoaML facilities are provided in the form of:

- Two SoaML diagram types - SoaML Component diagram and SoaML Sequence diagram - accessed through the 'New Diagram' dialog

- SoaML pages in the Diagram Toolbox
- SoaML element and relationship entries in the 'Toolbox Shortcut' menu and Quick Linker

Example SoaML Diagram



Notes

- Service Oriented Architecture Modeling Language (SoaML) is supported in the Corporate, Unified and Ultimate Editions of Enterprise Architect

SoaML Toolbox Pages

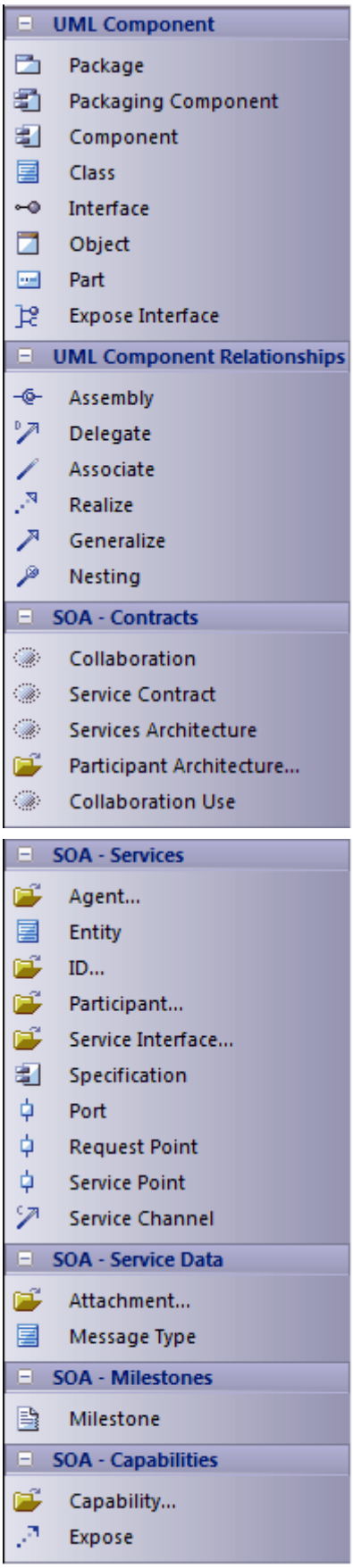
You can create the elements and relationships of the SoaML model using the 'SoaML' pages of the Diagram Toolbox. Each of the two SoaML diagram types has a separate set of pages, although the last five (SOA-specific) pages in the two sets are identical.

Access

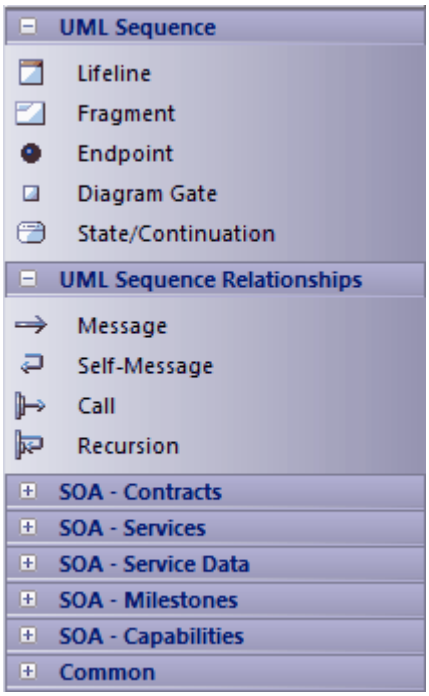
Ribbon	Design > Diagram > Toolbox : to display the 'Find Toolbox Item' dialog and specify 'SoaML Component' or 'SoaML Sequence'
Keyboard Shortcuts	Ctrl+Shift+3 : > Specify 'SoaML Component' or 'SoaML Sequence' in the 'Find Toolbox Item' dialog
Other	Diagram caption bar | Click the icon to display the Diagram Toolbox : > Specify 'SoaML Component' or 'SoaML Sequence' in the 'Find Toolbox Item' dialog

Toolbox Pages

SoaML Component Diagram Toolbox



SoaML Sequence Diagram Toolbox



SOMF 2.1

The Service-Oriented Modeling Framework (SOMF) is a service-oriented development life cycle methodology, offering a number of modeling practices and disciplines that contribute to successful service-oriented life cycle management and modeling. This text is derived from the extensive Wikipedia entry on Service Oriented Modeling:

'The Service-Oriented Modeling Framework (SOMF) has been proposed by author Michael Bell as a holistic and anthropomorphic modeling language for software development that employs disciplines and a universal language to provide tactical and strategic solutions to enterprise problems. The term "holistic language" pertains to a modeling language that can be employed to design any application, business and technological environment, either local or distributed. This universality may include design of application-level and enterprise-level solutions, including SOA landscapes or Cloud Computing environments. The term "anthropomorphic", on the other hand, affiliates the SOMF language with intuitiveness of implementation and simplicity of usage.'

'SOMF ... illustrates the major elements that identify the “what to do” aspects of a service development scheme. These are the modeling pillars that will enable practitioners to craft an effective project plan and to identify the milestones of a service-oriented initiative—either a small or large-scale business or a technological venture.'

SOMF in Enterprise Architect

In Enterprise Architect, SOMF 2.1 is implemented as a profile within an MDG Technology that is integrated with the Enterprise Architect installer. The SOMF 2.1 facilities are provided in the form of:

- Eleven SOMF diagram types, accessed through the 'New Diagram' dialog:
  - Conceptual
  - Analysis
  - Cloud Computing
  - Logical Design Relationship
  - Logical Design Composition
  - Business Integration
  - Conceptual Architecture
  - Asset Utilization
  - Transaction
  - Transaction Directory
  - Reference Architecture
- SOMF pages in the Toolbox - Enterprise Architect includes several Toolbox pages of modeling structures for each SOMF 2.1 diagram type, located through the Toolbox search facilities; these provide a wide breadth of SOMF modeling capabilities
- SOMF element and relationship entries in the Toolbox Shortcut menu and Quick Linker

National Information Exchange Modeling (NIEM) 2.1

National Information Exchange Modeling (NIEM) provides a common framework that is used to define how information can be shared between systems, government agencies and departments. The NIEM UML Profile helps you to:

- Create and develop UML-based Information Exchange Package Documentation (IEPD) models, either:
  - Generating an IEPD from an Enterprise Architect Pattern to produce all necessary exchange files, static artifacts, metadata and catalog files, or
  - Using the Schema Composer to generate your own NIEM subset namespaces, automatically detecting inter-dependencies, and using the resulting subset schema to build your own IEPD
- Create PIM, PSM and Model Package Description (MPD) diagrams, using the NIEM Toolbox pages
- Import NIEM Reference Schema into your model
- Generate NIEM Schema from your model

Create a NIEM IEPD Model from a Pattern

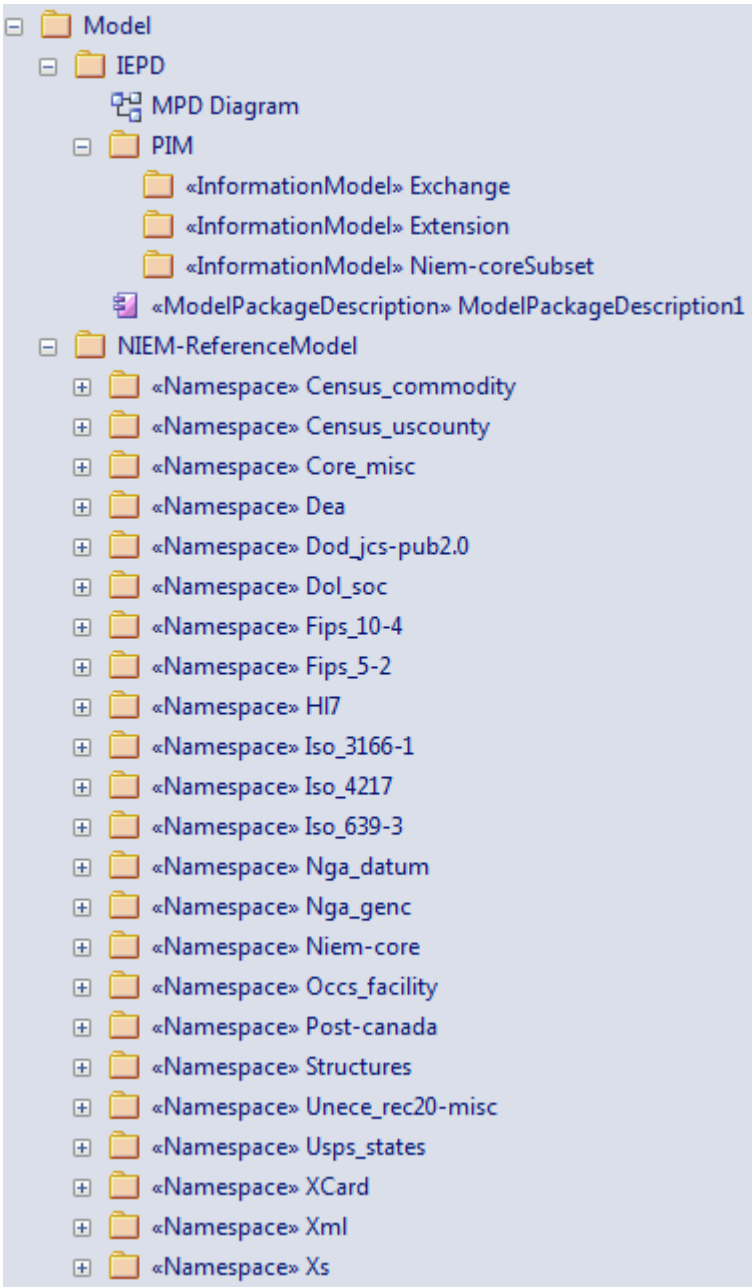
The NIEM UML Profile provides a model Pattern from which to build IEPD models. You can apply this Pattern in your NIEM project, using the Model Builder.

1. Open the Model Builder (Ctrl+Shift+M), then select 'Information Exchange > NIEM' in the '<<perspective>>'

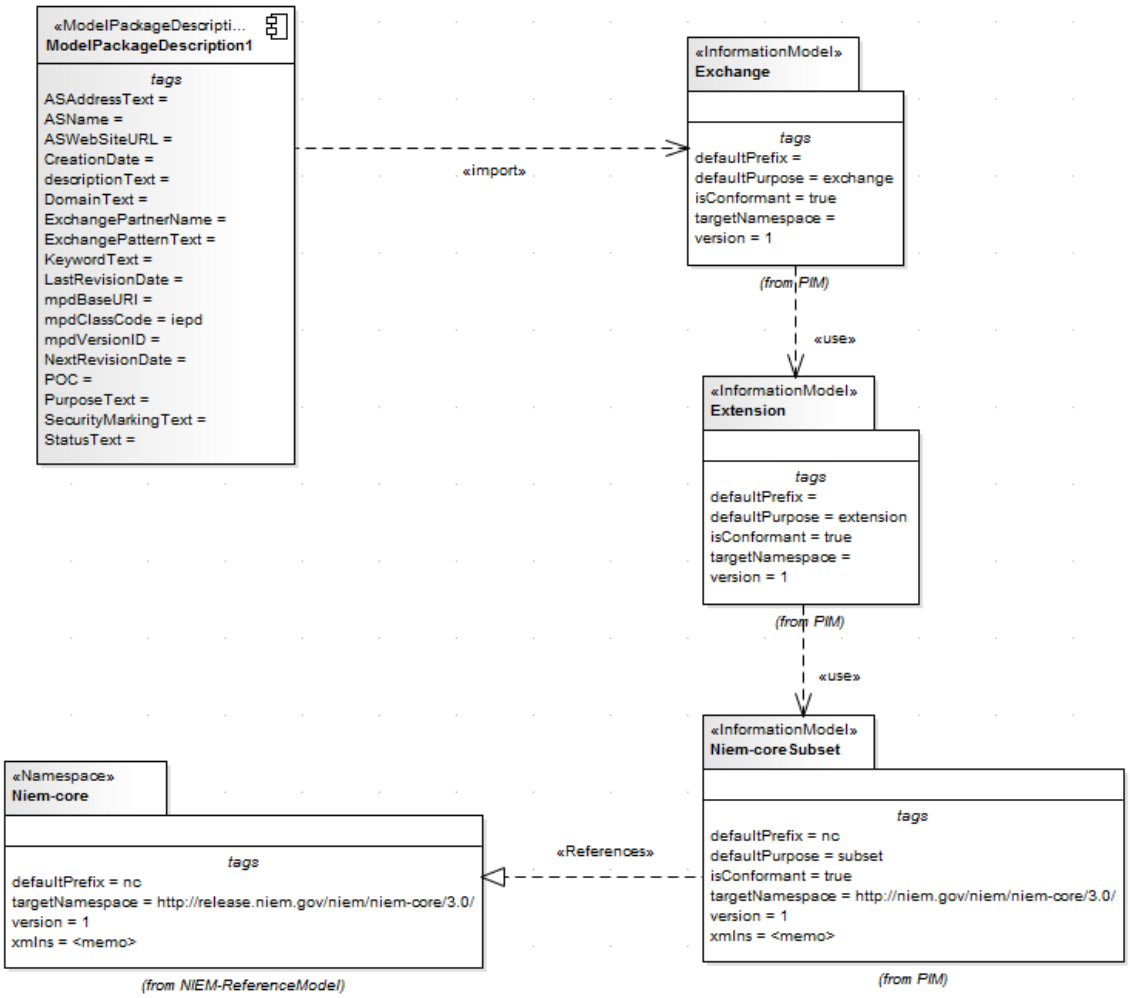
field.

2. In the 'NIEM 3 and 4' list, scroll through the technologies and click on the required 'NIEM *n.n* Reference Model', then Ctrl+Click on the corresponding 'NIEM *n* IEPD Starter Model'.
3. Click on the Create Model(s) button.

The system generates a new model containing an IEPD Package (itself containing a PIM Package), and a NIEM ReferenceModel Package. The Reference Model can take some time to download.



The IEPD Package contains a top-level Model Package Description (MPD) diagram (as shown), which contains the MPD Component and all the namespaces and files related to it.



The PIM Package consists of all the namespaces and subset namespaces for the IEPD. The relationships between the namespaces and the MPD Component is shown in the MPD diagram. The MPD Component must import at least one namespace for successful NIEM schema generation.

The NIEM ReferenceModel Package includes all the NIEM reference schema models for the selected NIEM version.

NIEM Diagrams

You can also create all of the appropriate diagrams from the NIEM diagram set and from the corresponding NIEM Diagram Toolbox pages. These diagrams are of three types:

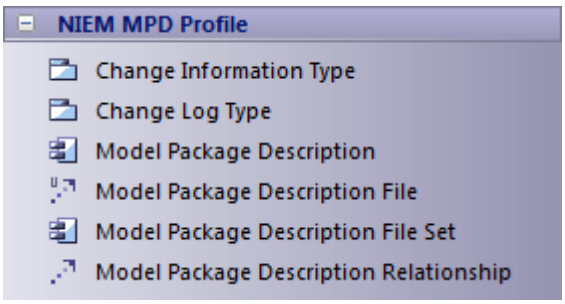
- NIEM Platform Independent Model (PIM) diagram
- NIEM Platform Specific Model (PSM) diagram
- NIEM Model Package Description (MPD) diagram

The templates from which to develop these diagrams are available through the 'New Diagram' dialog.

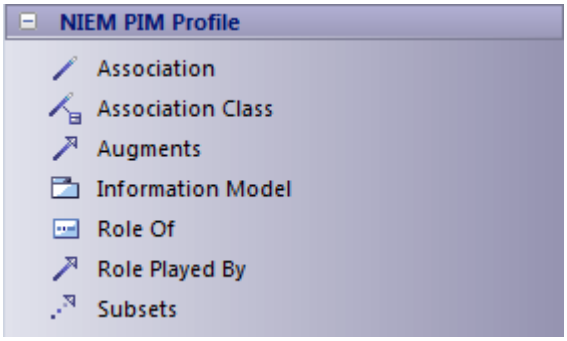
NIEM Toolbox Pages

Each diagram type has its own page of elements and connectors in the Diagram Toolbox. The NIEM UML Profile also provides a page of elements and connectors common to all three diagram types.

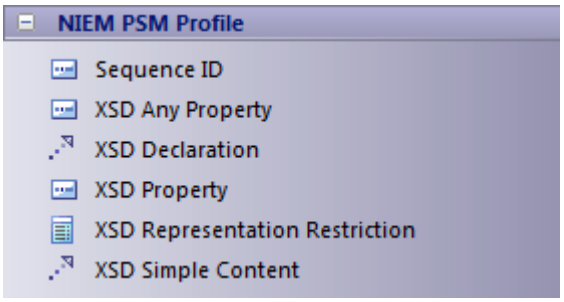
NIEM Model Package Description (MPD) Profile toolbox



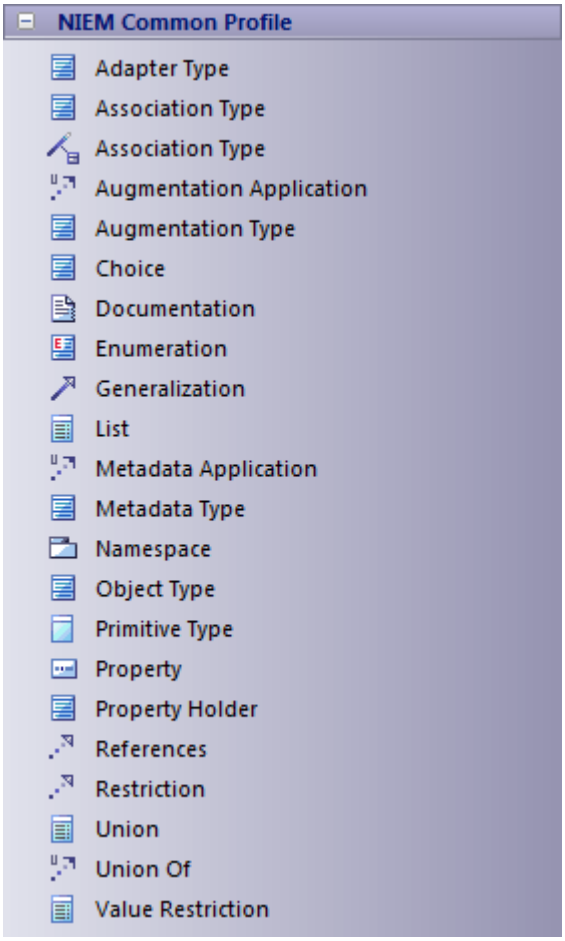
NIEM Platform Independent Model Toolbox



NIEM Platform Specific Toolbox



NIEM Common Profile Toolbox



Import NIEM Reference Schema

Step	Action
1	In the Browser window, right-click on the target Package and select the 'Specialize | NIEM 2.1 | Import NIEM 2.1 Schema' menu option.
2	On the 'Import XML Schema' dialog, in the 'Directory' field, type in or browse for the directory containing the schema to be imported, and select the

	.xsd schema files to import.
3	Under 'Import XSD Elements/Attributes as:' select the 'UML Attributes' radio button.
4	Click on the Import button. The NIEM model for the schema will be imported into the selected Package.

Generate NIEM Subset Namespaces

You can generate a subset namespace using the Enterprise Architect Schema Composer. This requires the NIEM Reference Model to be available in the model, as it is part of the IEPD Model Pattern.

Step	Action
1	Select the 'Develop > Schema Modeling > Schema Composer > Open Schema Composer' ribbon option.
2	Click on the New button to the right of the 'Profile' field. The 'New Message' dialog displays.
3	In the 'Name' field, type the name of the subset, and in the 'Namespace' field type the http address of the namespace.

4	In the 'Schema Set' field click on the drop-down arrow and select the 'National Information Exchange Model (NIEM)' option.
5	In the 'Save In' panel, select the 'Model Artifact' radio button.
6	Click on the icon and use the Navigator to select the namespace/information Model Package in IEPD | PIM, under which to generate the subset.
7	In the Browser window, open the NIEM ReferenceModel Package NIEM-core. Drag the Activity from this Package onto the left hand column of the Schema Composer. The attributes of this element are listed in the middle column of the Schema Composer.
8	Click on the checkbox for each of the attributes you require - for example, ActivityName and ActivityDateRepresentation. The corresponding Classes/NIEM object types are added to the left-hand column, whilst the right-hand column displays them as subset items.
9	Click on the Update button to save the subset profile. The status of the subset items displays against the item name in the left hand column and in the panel at

	the foot of the column.
10	Click on the Generate button. The 'Schema Export' dialog displays.
11	Select the checkbox against the items to generate, in the 'Technologies' panel. 'NIEM Model Subset' must be selected.
12	Click on the Generate button and, on the 'Find Package' dialog, select the namespace/ information model in which to generate the subset.
13	Click on the OK button, and on the second OK button. The subset model is generated.

NIEM Schema Generation

Once your NIEM IEPD model with its Extension information model, Exchange information model and Subset information model is complete, you can generate schema from it.

Step	Action
1	Right-click on the MPD Component, which imports the Exchange model, and select the 'Specialize|

	NIEM 2.1 | Generate NIEM 2.1 Schema' option. The 'Generate NIEM MPD Schemas' dialog displays.
2	In the 'Directory' field, type or browse for the directory path into which to generate the schema.
3	In the 'NIEM Version' field, click on the drop-down arrow and select the NIEM version for which to generate the schema. The static MPD artifacts and common artifacts (Catalog, Metadata) that will be generated are listed in the 'MPD Artifacts' panel, each with its relative path. The 'Namespace Schema(s)' panel shows the schema files that will be generated for the information models.
4	Click on the Generate button. Once the generation has completed successfully, click on the View Schema button to access the catalog file.

National Information Exchange Modeling (NIEM)

National Information Exchange Model (NIEM) provides a common framework that is used to define how information can be shared between systems, government agencies and organizations. Enterprise Architect's NIEM UML Profile helps you to:

- Create and develop UML-based Information Exchange Package Documentation (IEPD) models, by providing starter models, model Patterns and a number of Toolbox pages for creating IEPD models and schema models
- Generate complete IEPDs from your IEPD model
- Generate NIEM conformant schemas from your information models
- Import NIEM Reference Schema into your model
- Create NIEM subset namespaces, composed from elements of the NIEM Reference Schemas
- Create PIM, PSM and Model Package Description (MPD) diagrams, using the NIEM Toolbox pages

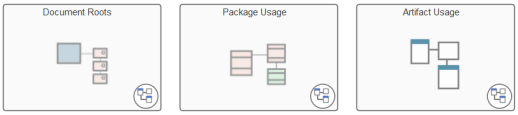
This illustration shows the NIEM 5.0 starter model, a Pattern provided as a part of NIEM in Enterprise Architect. (See the *Creating a NIEM IEPD* Help topic.)

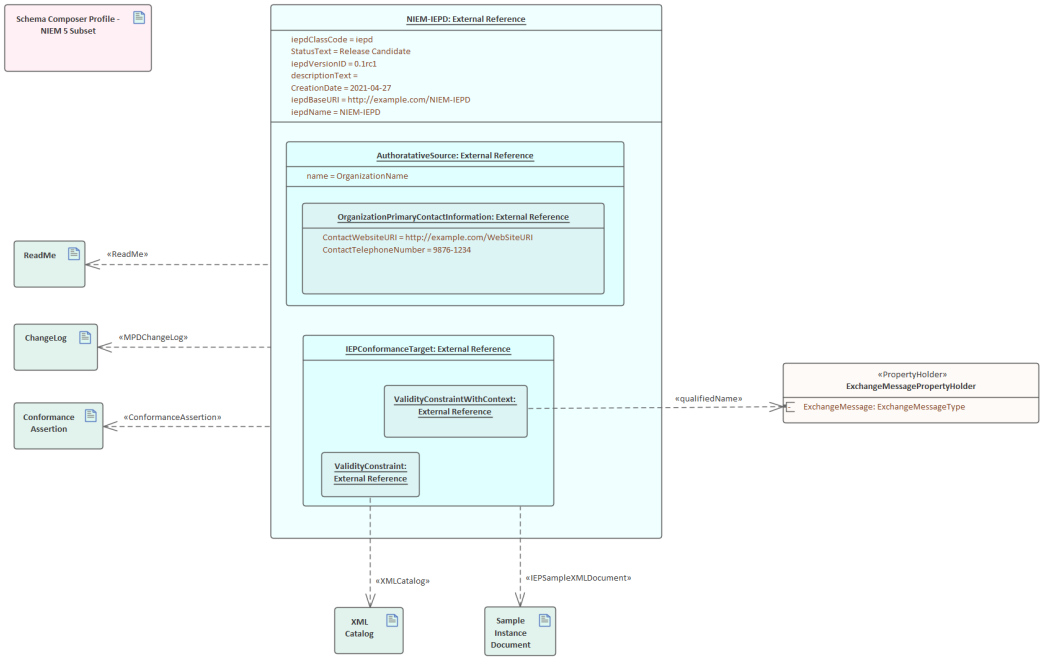
IEPD Overview

This IEPD Overview diagram provides a quick overview of what is contained in the NIEM Starter Model.

In the top-left corner is a Schema Composer profile artifact. You can double-click on this artifact to open the Schema Composer, pre-configured to create a NIEM subset schema.

The other diagrams in the Starter Model, can be opened using the buttons to the right. Each of these diagrams focus in on just one aspect of the IEDP model, giving you room to add additional items without causing the diagram to become cluttered.





UML Profile for NIEM

Enterprise Architect integrates with a UML Profile for NIEM (supporting NIEM 5, 4 and 3), along with a number of model Patterns to help you get started in modeling your NIEM project.

The profile defines a collection of stereotypes for use in building NIEM models. It also defines three different diagram types: Model Package Description (MPD) diagram, Platform Independent Model (PIM) diagram and Platform Specific Model (PSM) diagram. Each of these diagram types has corresponding Diagram Toolbox pages, from which you can select items to add to your model by dropping them onto a diagram.

Access

Use any of the methods outlined here to display the Diagram Toolbox, then click on to display the 'Find Toolbox Item' dialog and specify 'NIEM 3.0 MPD' (or 'PIM' or 'PSM').

The Diagram Toolbox that corresponds to a particular diagram type becomes active whenever you open a diagram of that type. However, you can also access any Diagram Toolbox at any time, using this method:

- From the top of the Diagram Toolbox, click on to display the 'Find Toolbox Item' dialog and specify '<profile> <toolbox>'

To reset the Toolbox to the default type for the current diagram, simply close then reopen the diagram.

Ribbon	Design > Diagram > Toolbox
Keyboard Shortcuts	Ctrl+Shift+3
Other	Click the » icon on the Diagram caption bar to display the Diagram Toolbox

Diagram Toolbox Pages

The NIEM Diagram Toolboxes provide quick access to the elements and connectors that you commonly use in a particular type of diagram.

The MPD Diagram Toolbox is grouped into a number of separate pages: Model Patterns, Relationships, File Type Usage and Schema Document Usage. The PIM and PSM diagrams share a common Toolbox page, as well as each having their own specific Toolbox page.

Common Toolbox Items

The NIEM Common Profile consists of stereotypes that are used in both the NIEM PIM Profile and the NIEM PSM Profile.

Icon	Description

AdapterType	A NIEM adapter type is a NIEM object type that adapts external components for use within NIEM.
AssociationType	A NIEM association type establishes a relationship between objects, along with the properties of that relationship.
AssociationType	A NIEM association type establishes a relationship between objects, along with the properties of that relationship.
AugmentationType	A NIEM augmentation type is a complex type that provides a reusable block of data that can be added to object types or association types.
Choice	A Choice Class groups a set of attributes whose values are mutually exclusive.
Documentation	A Documentation comment is the data definition of the element that owns it.
Generalization	A UML Generalization
List	A List is a DataType whose values consist of a finite length (possibly empty) sequence of values of another DataType,

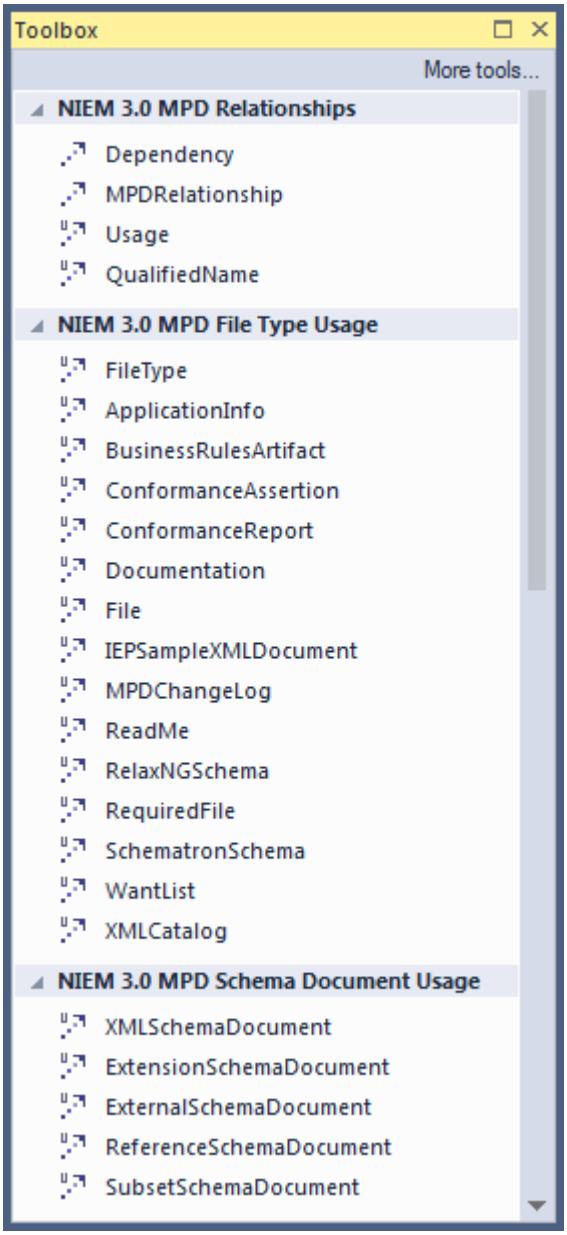
	which is the item type of the List.
LocalVocabulary	Local vocabulary defines a set of domain specific terms or abbreviations that then can be used in NIEM names and definitions.
LocalTerm	The LocalTerm stereotype defines a domain-specific word, phrase, acronym, or other string of characters used in a LocalVocabulary.
MetadataApplication	The «MetadataApplication» stereotype applies to a Usage between a «MetadataType» Class and either another «MetadataType» Class or a Property. It represents a constraint on a NIEM «MetadataType» that limits the application of the NIEM «MetadataType» to specific schema types or schema elements.
MetadataType	A NIEM metadata type describes data about data, that is, information that is not descriptive of objects and their relationships, but is descriptive of the data itself.
Namespace	A Namespace Package represents a

	NIEM namespace identified by a target namespace URI.
NIEMType	A NIEMType is a Class that represents one of the specific semantic kinds of NIEM complex types (that is, types that can have attributive structure). NIEMType is abstract.
ObjectType	A NIEM object type represents some kind of object: a thing with its own lifespan that has some existence.
PrimitiveType	The NIEM Primitive Type Library defines a predefined set of UML primitive types to be used in NIEM-UML models. To insure integrity and consistency of the type system used at the PIM level with the generation of NIEM compliant schema, the primitive types in this library are based on XML schema primitive types.
Property	
PropertyHolder	A PropertyHolder is a Class holding global Properties that are not the subject of any specific NIEM type. Property declarations of this kind define

	the object type of the property without restricting its use to a specific type of subject.
References	The References stereotype applies to a Realization between Properties, Classes or Packages. It allows for Properties in one Class to be defined by reference to Properties in another class.
Representation	The NIEM Representation Pattern, allows for a type to contain a representation element, and the various representations for that element type are in the substitution group for that representation element.
Restriction	A Restriction Realization represents a relationship between two type definitions: the first is derived by restriction from the second.
Union	A Union is a DataType whose value space is the union of one or more other DataTypes, which are the member types of the Union.
UnionOf	The UnionOf stereotype is applied to a Usage dependency, the client of which

	must be a Union DataType and the supplier of which must be a DataType that represents a legal union member type. A UnionOf dependency specifies that the supplier DataType is a member type of the client Union.
ValueRestriction	

NIEM 3.0 MPD Toolbox



MPD Toolbox Items

The Model Package Description Profile comprises stereotypes and artifacts that are used to model NIEM MPDs.

Icon	Description
Relationships	

Dependency	A UML Dependency relationship.
MPDRelationship	The ModelPackageDescriptionRelationship stereotype applies to a Dependency that represents a relationship between MPDs or between an MPD and another resource (such as a NIEM specification; as in the case of conforms-to).
Usage	A UML usage relationship
QualifiedName	The <<qualifiedName>> Usage connector is used to specify the Document Element of an IEP. To identify a Document Element of an IEP in UML: • Add an instance of IEPConformanceTargetType to the IEPConformanceTarget slot of the ModelPackageDescription Artifact instance • Add a QualifiedNamesType instance to the ValidityConstraintWithContext slot of the new IEPConformanceTargetType instance • Add a Usage with applied Stereotype ?qualifiedName? where the client is the

	new QualifedNamesType instance and the supplier is a Property representing an XSD Element
File Type Usage	
FileType	The <<FileType>> Usage connector is a data type for describing an MPD file artifact. It is also the base type from which many other <<FileType>> Usage connectors are derived.
ApplicationInfo	The <<ApplicationInfo>> connector extends the <<FileType>> usage connector. It is used to specify an MPD artifact that is used by a software tool (for example, import, export, input and output).
BusinessRulesArtifact	The <<BusinessRulesArtifact>> connector extends the <<FileType>> usage connector. It is used to specify an MPD artifact that contains business rules and constraints on exchange content.
ConformanceAssertion	The <<ConformanceAssertion>> connector extends the <<FileType>> usage connector. It is used to specify an

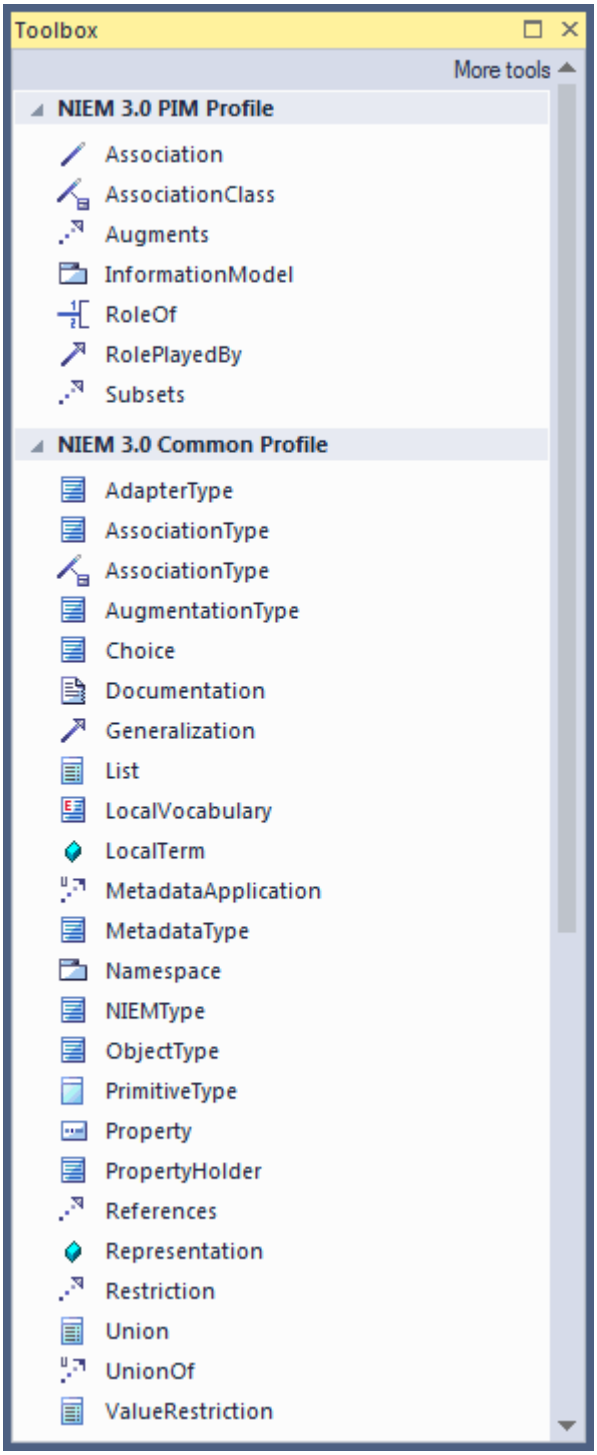
	MPD artifact that represents a declaration that a NIEM IEPD is NIEM-conformant.
Conformance Report	The <<ConformanceReport>> connector extends the <<FileType>> usage connector. It is used to specify an MPD artifact either auto-generated by a NIEM-aware software tool or manually prepared that checks NIEM conformance and/or quality and renders a detailed report of results.
Documentation	The <<Documentation>> connector extends the <<FileType>> usage connector. It is used to specify an MPD artifact that is a form of explanatory documentation.
File	The <<File>> connector extends the <<FileType>> usage connector. It is used to specify a generic electronic file artifact in an MPD; a file stored on a computer system.
IEPSampleXMLDocument	The <<IEPSampleXMLDocument>> connector extends the <<FileType>> usage connector. It is used to specify an example MPD instance XML document or IEP artifact.

MPDChangeLog	The <<MPDChangeLog>> connector extends the <<FileType>> usage connector. It is used to specify an MPD artifact that contains a record of the MPD changes.
ReadMe	The <<ReadMe>> connector extends the <<FileType>> usage connector. It is used to specify an MPD read-me artifact.
RelaxNGSchema	The <<RelaxNG>> connector extends the <<FileType>> usage connector. It is used to specify a RelaxNG schema.
RequiredFile	The <<RequiredFile>> connector extends the <<FileType>> usage connector. It is used to specify an MPD file artifact that another artifact depends on and should not be separated from.
SchematronSchema	The <<SchematronSchema>> connector extends the <<FileType>> usage connector. It is used to specify a Schematron schema document.
WantList	The <<WantList>> connector extends the <<FileType>> usage connector. It is used to specify an MPD artifact that represents

	a NIEM schema subset and is used as an import or export for the NIEM SSGT.
XMLCatalog	The <<XMLCatalog>> connector extends the <<FileType>> usage connector. It is used to specify an MPD artifact that is an OASIS XML catalog.
Schema Document Usage	
XMLSchema Document	The <<XMLSchemaDocument>> connector extends the <<FileType>> usage connector. It is used to specify an MPD artifact that is an XML schema document (that is, an XSD that is not necessarily a NIEM subset, extension, or reference schema).
ExtensionSchemaDocument	The <<ExtensionSchemaDocument>> connector extends the <<XMLSchemaDocument>> usage connector. It is used to specify an MPD artifact that is a NIEM extension schema document.
ExternalSchemaDocument	The <<ExternalSchemaDocument>> connector extends the

	<<XMLSchemaDocument>> usage connector. It is used to specify an MPD artifact that is a schema document external to NIEM.
ReferenceSchemaDocument	The <<ReferenceSchemaDocument>> connector extends the <<XMLSchemaDocument>> usage connector. It is used to specify an MPD artifact that is a reference schema document (from a release, domain update, or core update).
SubsetSchemaDocument	The <<SubsetSchemaDocument>> connector extends the <<XMLSchemaDocument>> usage connector. It is used to specify an MPD artifact that is a subset schema document.

NIEM 3.0 PIM Toolbox



PIM Toolbox Items

The NIEM PIM Profile comprises stereotypes that are used in NIEM PIMs but not NIEM PSMs.

Icon	Description
Association	A UML Association.
AssociationClass	A UML Association Class.
Augments	A stereotyped Realization connector, used to specify that one Class (the supplier) Augments another Class (the client).
Information Model	InformationModel is a stereotyped Package that provides a platform-independent perspective on the structure of information to be exchanged in NIEM messages. It represents a NIEM namespace, but can also specify a default purpose, such as subset, exchange or extension.
RoleOf	The RoleOf stereotype is applied to an AssociationEnd to specify the type of the role of the associated property.
RolePlayedBy	A stereotyped Generalization connector specifying that the role played by instances of the general Class will be the type of the special Class.

Subsets	The Subsets connector is a stereotyped realization that specifies a subset relationship between a subset client (the derived element) and its reference supplier (the base element).

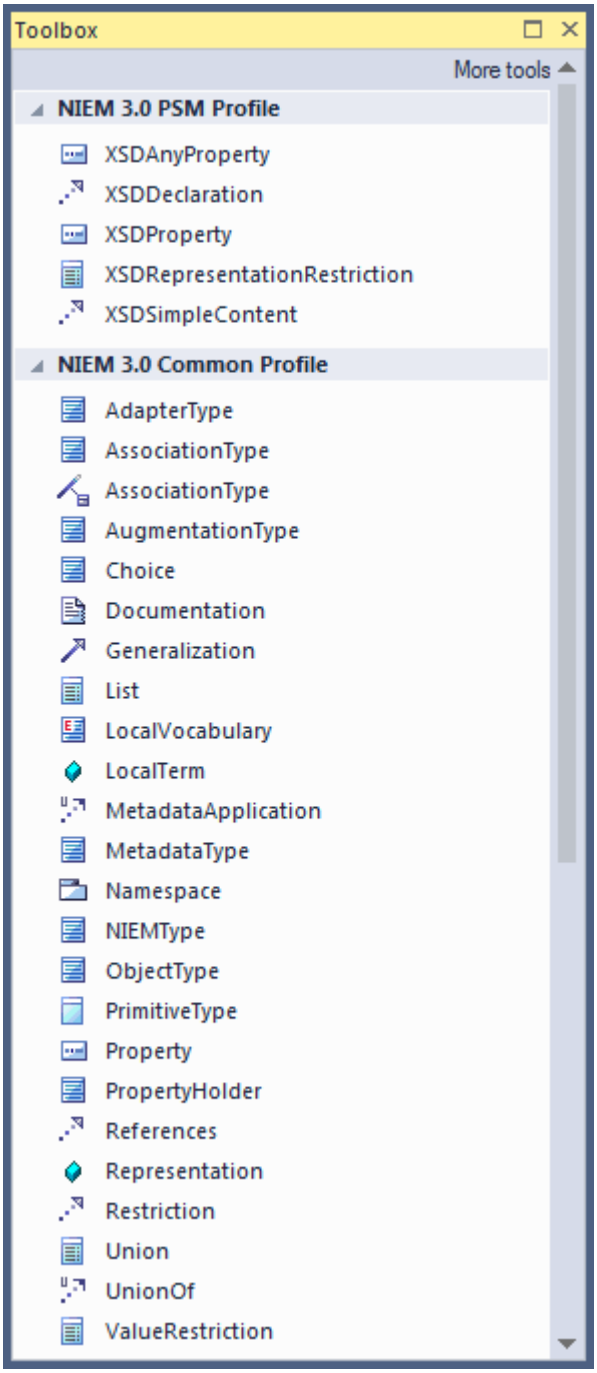
PSM Toolbox Items

The NIEM PSM Profile comprises stereotypes that are used in NIEM PSMs. These stereotypes need not be used with a NIEM PIM, but they can be in order to provide additional platform-specific markup.

Icon	Description
XSDAnyProperty	XSDAnyProperty stereotype represents a property that is unrestricted with respect to its type, which is implemented in XML Schema as the xs:any particle.
XSDDeclaration	The XSDDeclaration stereotype is a specialization of the common References stereotype.
XSDProperty	An XSDProperty Property represents a NIEM property, which is implemented in XML Schema as either an attribute declaration and use or an element

	declaration and particle
XSDRepresentationRestriction	XSDRepresentationRestriction specifies a restriction on the representation in an XML schema of the values of a base DataType.
XSDSimpleContent	The «XSDSimpleContent» stereotype represents a relationship between two type definitions: the first is a complex type definition with simple content, the second is a simple type.

NIEM 3.0 PSM Toolbox


Toolbox
More tools
NIEM 3.0 PSM Profile
XSDAnyProperty
XSDDeclaration
XSDProperty
XSDRepresentationRestriction
XSDSimpleContent
NIEM 3.0 Common Profile
AdapterType
AssociationType
AssociationType
AugmentationType
Choice
Documentation
Generalization
List
LocalVocabulary
LocalTerm
MetadataApplication
MetadataType
Namespace
NIEMType
ObjectType
PrimitiveType
Property
PropertyHolder
References
Representation
Restriction
Union
UnionOf
ValueRestriction

Download the NIEM Reference Model

The NIEM 5 Reference Model is a UML representation of the content of the NIEM 5 Release Package XSD files.

It contains Packages representing NIEM-core, as well as the various Domain schemas included in the NIEM 5 release, their associated Codes lists and other associated Packages. The NIEM 5 reference model is available for download into your Enterprise Architect project, from the Sparx Systems Reusable Asset Server.

Earlier versions of NIEM reference models are also available for download from the Sparx Systems Reusable Asset Server.

Access

Display the Model Builder using any of the methods outlined here.

In the Model Builder dialog, select the Perspective "Information Exchange > NIEM" and then select 'NIEM 3, 4 and 5'.

Select a Reference Model, MPD Types and Starter Model, as required.

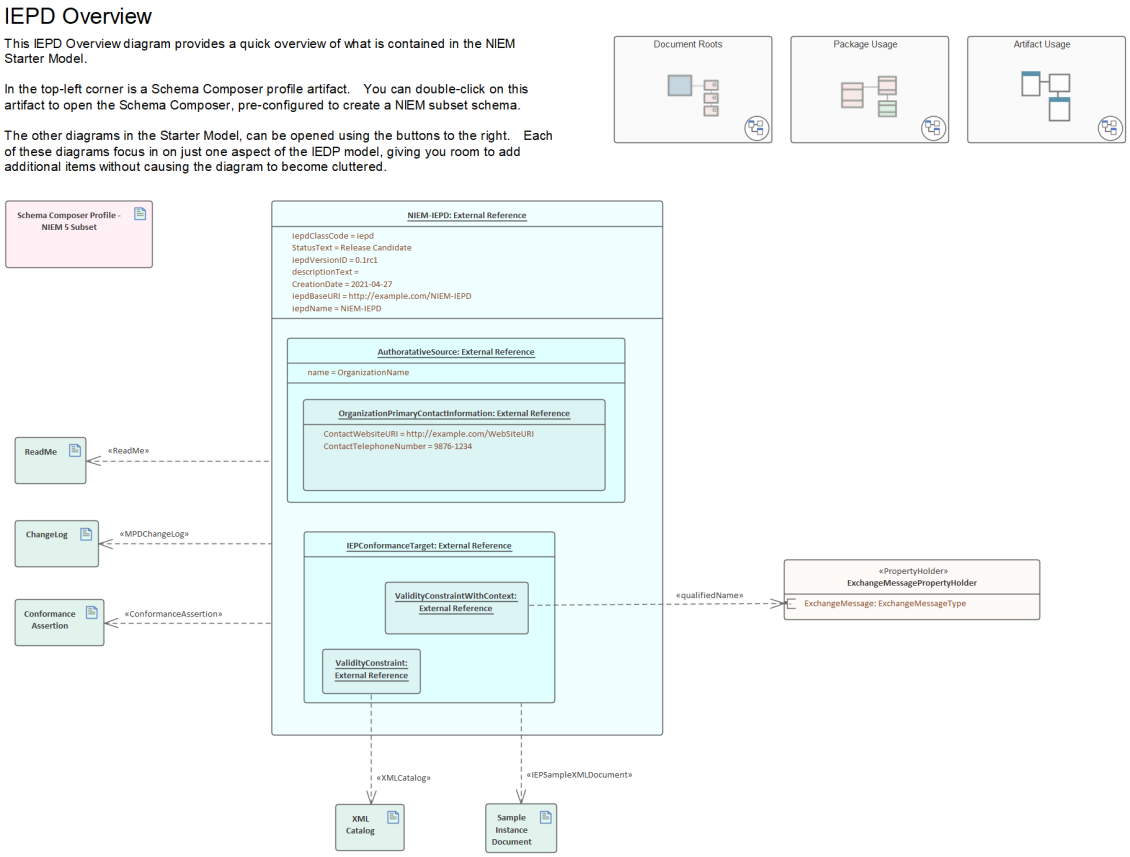
Ribbon	Start > Personal > Model Builder Design > Package > Model Builder
Context	Right-click on Package | Model Builder

Menu	(pattern library)
Keyboard Shortcuts	Ctrl+Shift+M
Other	Browser window caption bar menu | Model Builder (pattern library)

Creating a NIEM IEPD

Enterprise Architect's NIEM profile provides a basic IEPD model as a starting point from which you can build your own IEPD model.

You can add the IEPD starter model to your project using the Model Builder.



The IEPD Starter Model Pattern, available from the Model Builder.

This topic provides an overview of the steps required to create a new NIEM IEPD model in Enterprise Architect, and to generate an IEPD from that model.

Creating a NIEM IEPD model and generating a NIEM IEPD

Steps	Description
Create a new Enterprise Architect project	Launch Enterprise Architect and create a new project. From the ribbon choose "Design > Package > Model Builder". In the Model Builder dialog, click on the <perspective name> button and select 'Information Exchange | NIEM' and expand the 'NIEM 3, 4 and 5' Perspective. It is essential that your NIEM project contains the NIEM IEPD Types and at least one of the NIEM Reference Models. Select the 'NIEM 5.0 Reference Model' as well as 'NIEM 5 IEPD Types'. Click on the Create Model button to download and import the selected models into your project. Also available in the Model Builder is a model pattern for a basic NIEM IEPD. This is intended as a starting point for your NIEM project. Optionally, select the 'NIEM 5 IEPD Starter Model' and click on the Create

	Model button.
Create an IEPD model	If you chose not to include the IEPD model in the previous step, you can create your own model now. In the Browser window, create a new Package or (View node) to hold your IEPD model. Within the new Package, create a NIEM IEPD diagram. You can add instances of the types available in the NIEM IEPD Types Package to your diagram (and to your IEPD model), by dragging them onto your IEPD diagram. Use the Browser window to locate the Class types that you require, then press Ctrl as you drag the element into position on your diagram. The system prompts you to choose an action; either: • Place a link to the Class on the diagram, or • Create and add a new instance specification of the Class For the IEPD model, you would generally use Object Instances. To begin with, create an object instance of the IEPD Class. (As you will see

	shortly, you need an instance of the IEPD class to drive the generation of your IEPD.)
Customize your model	The instance of the IEPD class carries information that identifies the IEPD. The links between it and the various other model artifacts determine what is generated (and where it is generated to) when generating the schema files and catalog files. Whether you choose to download the IEPD Starter Model, or create your own IEPD model by dragging instances from the Browser window, you must set values for the properties of the Object instances that are appropriate to the model you are creating. This is achieved by setting the run-state properties of the various object instances used in your IEPD model. For detailed information on setting values for the IEPD instance objects, see the Help topic *Customize Your IEPD Model*
Create your data model	This is where you model the data that will be sent in your information exchange message. In NIEM, this is typically modeled within Packages that have the

	<<InformationModel>> stereotype, representing the different namespaces used in the model. These Packages typically include a NIEM-core Package that is a subset of the NIEM-core Reference Model Package, and two extension Packages that extend what is available from NIEM-core, one of which represents the exchange message. Your project might also require subsets of other NIEM schemas, such as those from the Biometrics or EmergencyManagement domains. For further information on creating data models, see the Help topics *Creating a NIEM Data Model* and *Subsetting NIEM with the Schema Composer*.
Generate the IEPD	Your NIEM model does not have to be complete before you generate an IEPD from it. Generating the IEPD can be considered an iterative process. You might perform a generation of just your namespace schemas before you have completed your IEPD, and before you have defined your conformance targets. You might generate with a fully described IEPD instance and conformance targets before you have

	defined your Information Models. You can continue to update your model and generate your IEPD however you see fit. To generate your IEPD, select the IEPD instance specification, either on the diagram or in the Browser window. Go to the 'Specialize' ribbon and select the option 'Technologies > NIEM > Generate NIEM Schema'. The Generate NIEM IEPD Schemas window opens. This window lists the Namespace Schemas that are used in your model, and you can select which of these to generate. You can also choose which of the NIEM infrastructure schemas to include in the generation. In this window you can also set the root directory for generation of the output files. Once you have made your selection and specified the output folder, click on the Generate button to commence generation of the IEPD. For detailed information on the Generate NIEM IEPD Schemas window, see the Help topic *NIEM IEPD Generation*.

Notes

- ALL projects containing NIEM models must include the NIEM IEPD Types Package downloaded via the Model Builder; the IEPD instance is central to your NIEM model The instance and the relationships to <<InformationModel>> Packages and other artifacts are used to drive the IEPD Generation; without an IEPD instance in your model, you will not be able to generate an IEPD
- Usually, you must have at least one of the NIEM Reference Models imported into your project; the reference models contain UML representations of the NIEM-core reference schema, as well as the many domain specific reference schemas, which must be available in your project if you intend to create subset schemas using Enterprise Architect's Schema Composer

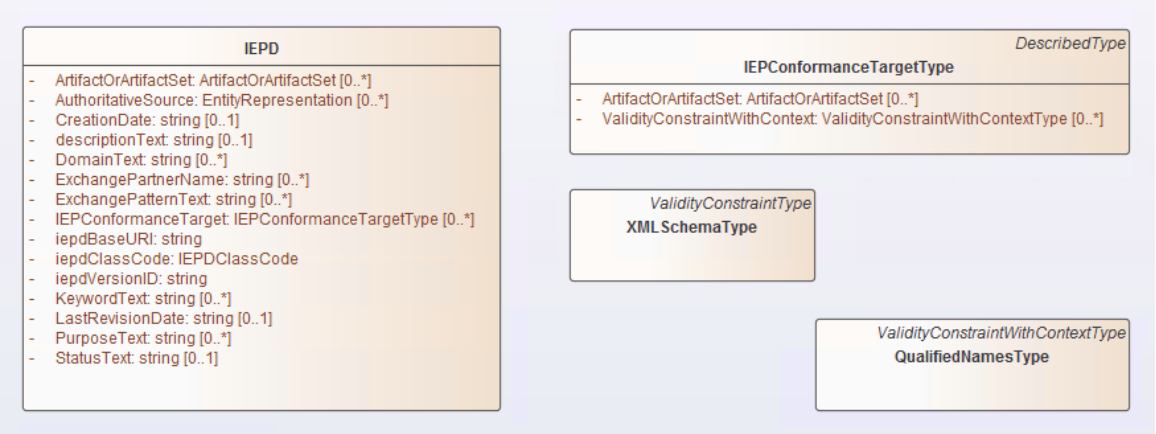
Customize Your IEPD Model

Whether you choose to download the IEPD Starter Model, or create your own IEPD model by dragging instances from the Browser window, you must set values for the properties of the Object instances that are appropriate to the model you are creating. This is achieved by setting the run-state properties of the various object instances used in your IEPD model.

NIEM IEPD Types

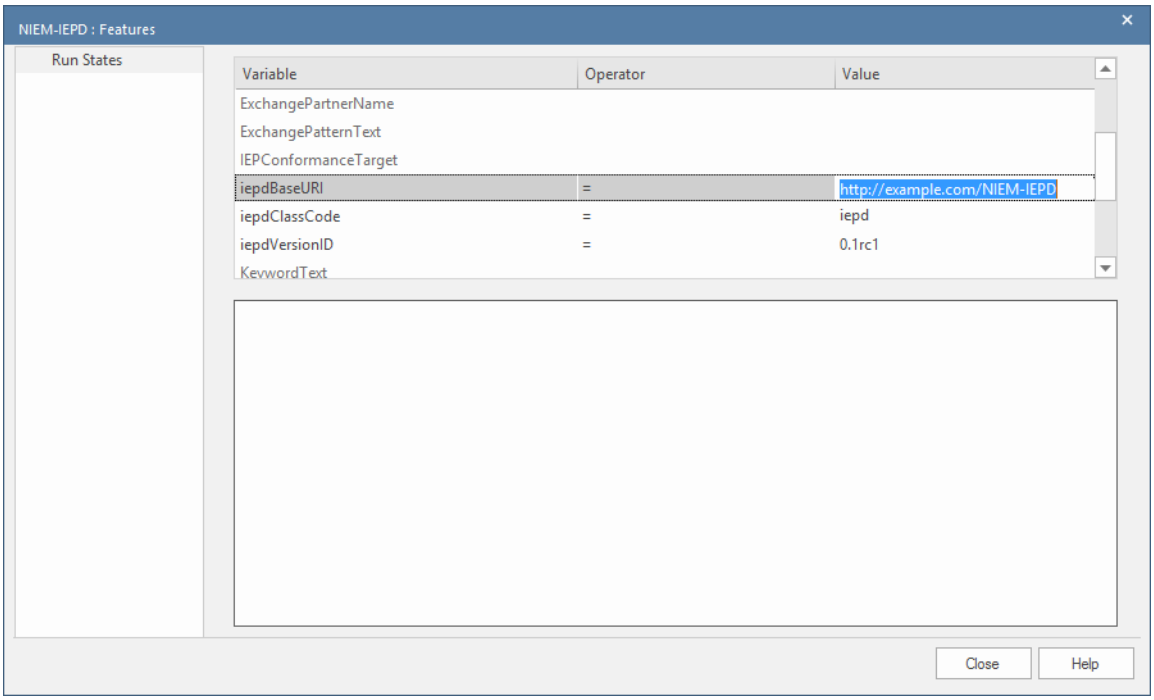
The Package 'NIEM IEDP Types' contains definitions for the Class 'IEPD', as well as a number of other Classes. These other Classes are referenced as classifiers for attributes of the 'IEPD' Class. Relationships between the various Classes defined in this Package can be viewed on the diagram 'NIEM-UML IEPD Types'.

The 'IEPD' Class has a number of attributes that are simple string types, as well as some attributes that are classified by types that are defined within the 'NIEM IEDP Types' Package.



Setting the Run-States of IEPD Objects

When setting the run state values for properties with simple types in the IEPD object, you can use the 'Set Run State' command. This can be accessed by right-clicking the Object on a diagram, then choosing 'Features | Set Run State...' (or by pressing Ctrl+Shift+R).

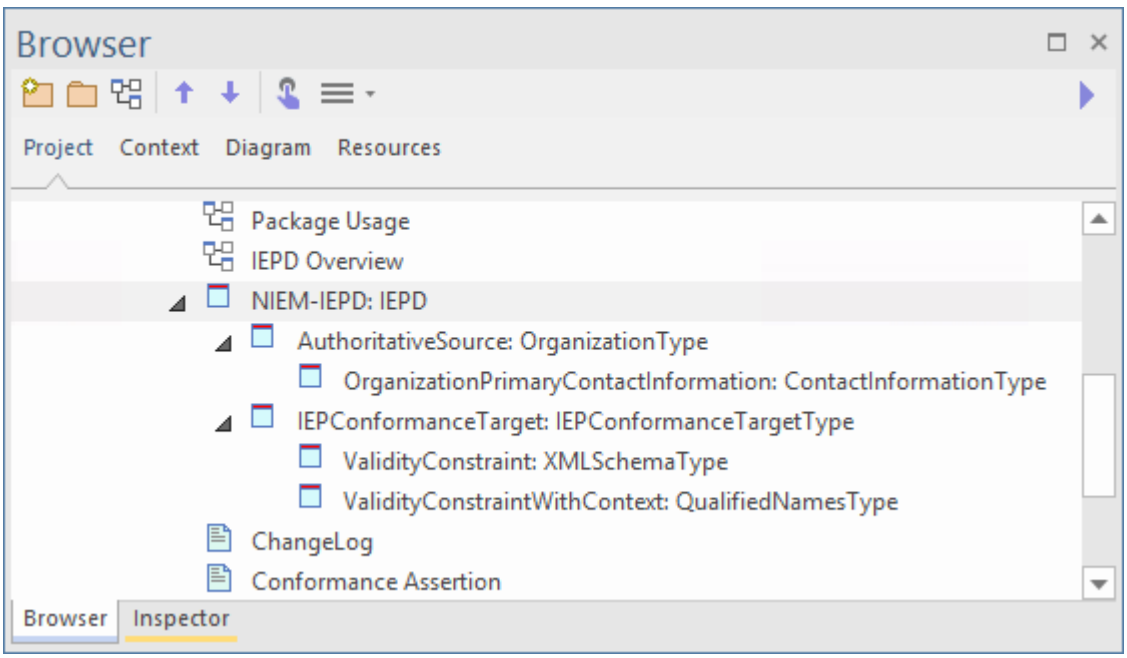


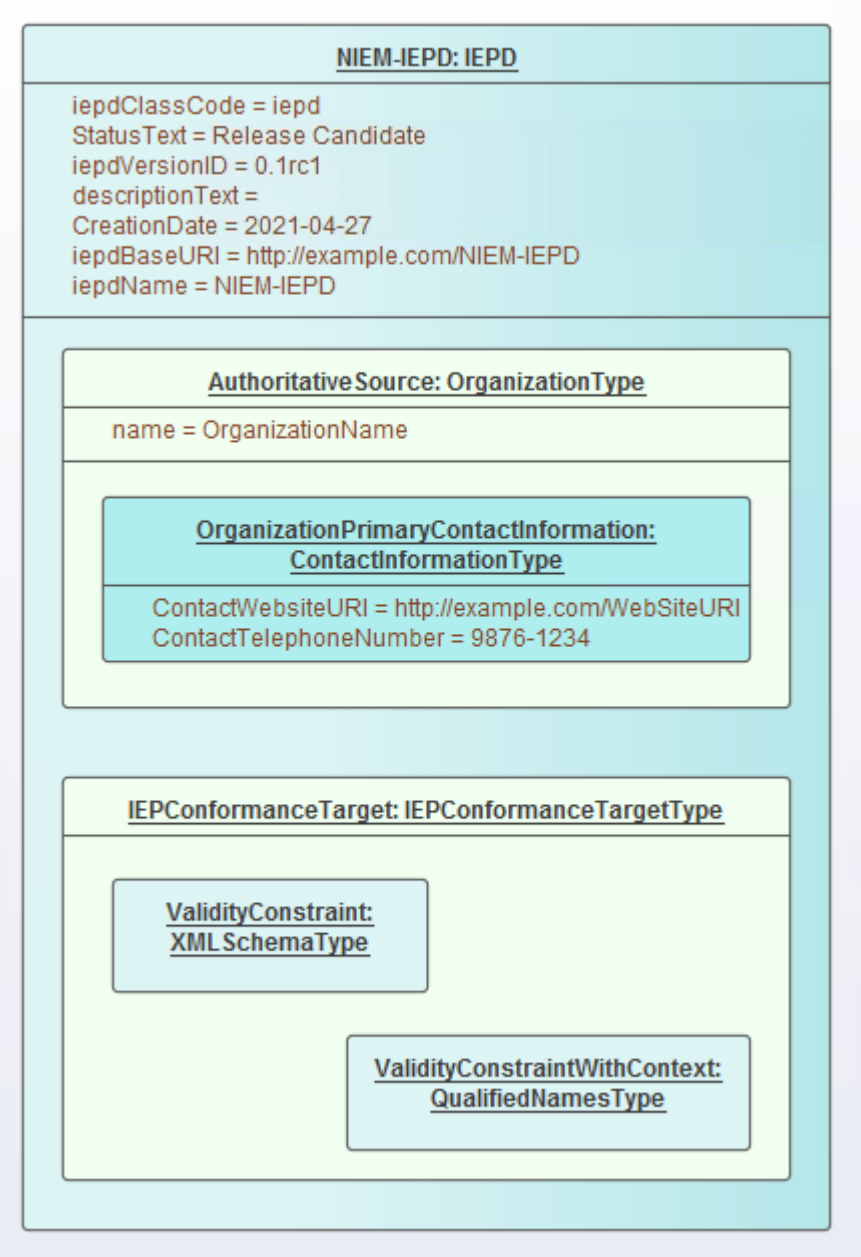
Where properties reference other Classes as their types, you cannot simply enter a run state value.

Enterprise Architect supports two methods of specifying values for these properties, each method requires creation of an Object instance of the referenced Class.

You should create an Object instance of the type corresponding to the property, then either create an Association between the two objects and set a role name for the property being set, or nest the Object as a child within the Object whose properties are being set and name the child Object using the name of the property being set. When associating an Object, the Object's name is not important, but the role name must match the name of the property being set.

For example, you might create an Object instance of the type IEPConformanceTargetType and nest it within the IEPD Object. In this case, the child Object must be named 'IEPConformanceTarget' to correspond with the attribute of that name in the Class definition. Ensure that the child Object is indeed nested within the parent, by inspecting the hierarchy shown in the Browser window.

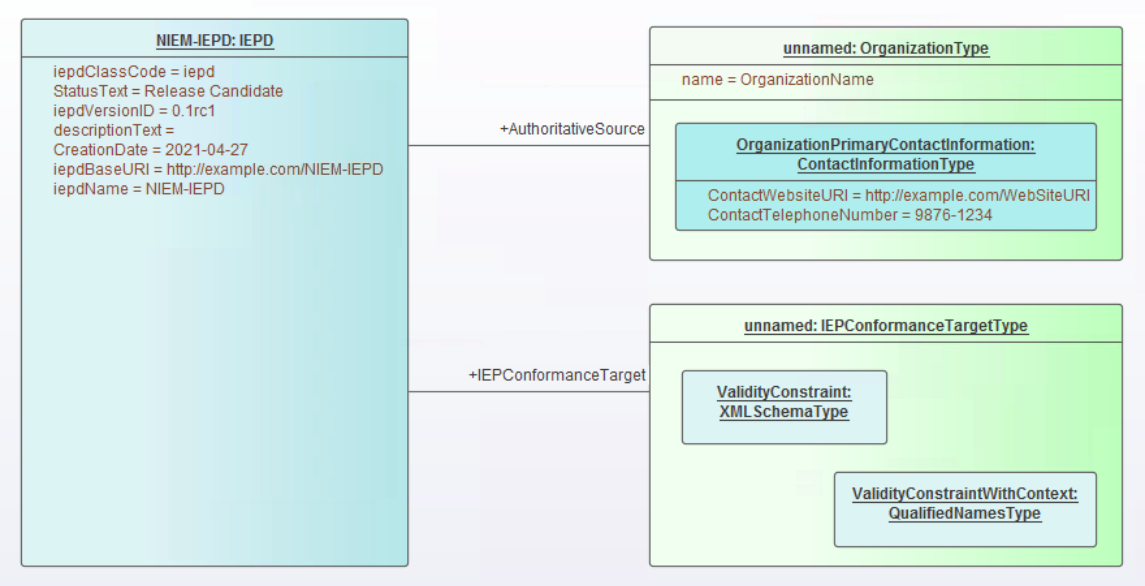




If using a role name on an Association, create the 'property' Object as a separate (non-nested) Object instance, then create an Association from the 'owner' to the 'object' and finally specify a role name for the target Object. For example, create an Association from the Model Package Description Object to an Object instance of IEPConformanceTargetType. Open the 'Properties' dialog for the Association and name the role of the target as 'IEPConformanceTarget', to correspond to the name of the

attribute in the 'IEPD' Class. Again, in this scenario, the name of the Object itself is not important, it could even be anonymous, but the role name must match the name of the attribute whose value you are setting.

Note that an IEPD Object might specify many IEPConformanceTargets. You must create an Object instance for each one and each one must be named 'IEPConformanceTarget'.

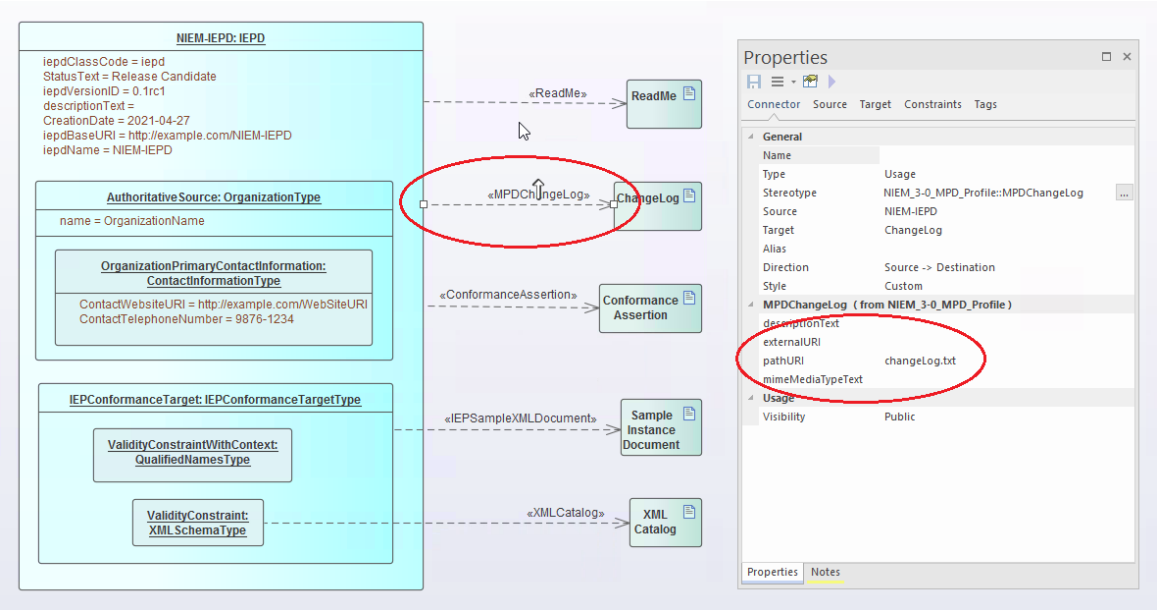


Either of these techniques can then be used to set properties within the IEPConformanceTargetType Object. For example, to set the value of the ValidityConstraintWithContext attribute, create an Object instance of the Class ValidityConstraintWithContextType (which might be an instance of the derived type QualifiedNamesType) and either name it and nest it, or associate it and name the role.

Modeling File Usage

File use can be modeled by adding Artifact elements to the diagram and linking with the required File Type Usage connector from the toolbox.

The various filenames are generated into the IEPD's XML catalog files, using values that are specified in the Properties of the relevant Usage connectors.



NIEM IEPD Generation

Generating the IEPD can be considered an iterative process. You do not have to wait until your NIEM model is complete before you generate an IEPD from it.

Your NIEM 'IEPD OVerview' diagram should contain an instance specification of an IEPD. The IEPD instance and its relationships to Conformance Target instances as well as other artifacts, is a representation of the IEPD catalog. When you generate an IEPD from your model, Enterprise Architect will generate a catalog file and other artifacts, based on the items in your IEPD model. It will also generate NIEM schemas for the <<InformationModel>> Packages referenced by your model. The result will be a collection of files output into the directory you specify for the generation process.

Steps for generating an IEPD

Step	Action
1	Your NIEM IEPD diagram should contain an Instance Specification of an IEPD. Select the IEPD instance, either on the diagram or in the Browser window.
2	From the 'Specialize' ribbon, choose the option

	'Technologies > NIEM > Generate NIEM Schema'. The 'Generate NIEM IEPD Schemas' dialog displays.
3	In the 'Directory' field, type or browse for the directory path into which to generate the IEPD.
4	The 'NIEM Version' field defaults to '5.0'. If generating a NIEM 3 or 4 IEPD, set this field to the appropriate value. The 'IEPD Artifacts' panel lists the static IEPD Artifacts and common Artifacts (such as Structures and Catalog) used in this model, each with its relative path. Select or deselect the checkboxes beside these items to generate or skip these items. The 'Namespace Schema(s)' panel shows the schema files that will be generated for the information models. Select or deselect the checkbox beside a Namespace Schema to generate or skip that schema. Select a Namespace Schema to display Package details for that schema.
5	Click on the Generate button. Once the generation has completed successfully, clicking the View Schema button opens Windows Explorer, showing the contents of the output

	directory used for generation. If the Catalog Artifact has been deselected, then clicking the View Schema button will open an editor to view the schema file associated with the currently selected Namespace Package.

Notes

- The output location of the schema file generated for a Package is specified by the 'pathURI' tag value on the Usage connector that relates the Package to the IEPD instance specification; default values are set by the Schema Composer when the subset Packages are created, but the values can be overridden by the user

Creating a NIEM Data Model

One of the underlying principles behind NIEM is re-use of a common reference vocabulary - a predefined set of data elements and definitions used to define information exchanges. To this end, one of the core tasks in building a NIEM data model is creating a subset of the NIEM reference schema. The goal is to model as much of your data exchange as is practical, by re-using types and elements that are already defined in the NIEM reference model.

A NIEM data model usually consists of a number of Packages with the <<InformationModel>> stereotype applied.

Typically, a model will have one Package representing a NIEM-core subset schema, some other Packages representing subsets of particular Domain schemas, and one or more Packages representing extension schemas. The extension schema Packages provide those elements required by the model that are not available from the NIEM Reference Model. Often, the root element of the exchange message is separated from more general elements, and modeled in an extension schema Package dedicated to the specific exchange.

Steps for Creating a NIEM Data Model

Step	Detail

Import the NIEM Reference Model	Many of the activities involved in creating NIEM models, rely on using the NIEM Reference Model. If you haven't already done so, import the Reference Model into your Enterprise Architect project before proceeding further. For more information, see the Help topic Download the NIEM Reference Model.
Create a Subset of the NIEM-core Reference Package	There are a number of reasons for creating subsets of the NIEM namespace schemas when creating NIEM IEPDs, but the two most important reasons are: • The reference schemas are very large; subsetting produces much smaller schema files that, in turn, leads to faster validation of the schemas • Elements within the reference schemas are very loosely constrained; the subsetting process allows modelers to impose much tighter constraints, such as restricting cardinality and allowed values, to more closely reflect actual business requirements In Enterprise Architect, the subsetting process is performed using the Schema Composer. The Schema Composer allows the modeler to select the subset of required

	Classes from the source Package and, for each of the selected Classes, select a subset of required attributes. The selected Classes with their reduced attribute sets are then copied to a target Package. Most often the source Package will be the NIEM-core namespace Package from the NIEM Reference Model. In this case, the target Package will also be a namespace Package named 'NIEM-core', but it will be part of your NIEM IEPD model. Other namespace Packages from the Reference Model, such as the Domain Packages, can also be subsetted in the same way. Use Enterprise Architect's Schema Composer tool to copy a subset of the NIEM-core reference Package to the NIEM-core subset Package that is part of your IEPD model. The goal is to model as much of your data exchange as is practical, by re-using types and elements that are already defined in the NIEM-core Reference Model. In cases where your model will also make use of NIEM Domain Packages, this subsetting process should be repeated for each domain Package that you use. For more information, see the Help topic

	Subsetting NIEM with the Schema Composer.
Create Extension Packages	When creating a NIEM data model, the aim is to model as much of your data exchange as possible using types and elements from the NIEM Reference Model. What cannot be modeled by re-using existing NIEM elements is then modeled in 'extension' namespace Packages, by creating new types and elements using elements from the NIEM-UML profiles, with all types ultimately deriving from XML schema primitive types. Both the NIEM Starter Model (from the Model Builder) and the IEPD Starter Model Pattern (from the Diagram Toolbox) provide <<InformationModel>> Packages in which to model the various schemas. Using PIM diagrams within these Packages, you can build models of your different schemas, by adding elements from the Diagram Toolbox. It is suggested that you use the diagram in the 'exchange' Package, to assemble the high-level model of your exchange, using types and elements from other schema

	Packages as required. Most IEPDs require extension schemas to define specific types and properties that are unique to the data exchange being defined. However, the NIEM model does not define specific message types or structures for assembling all of the objects in an exchange. It is therefore up to the creator of the IEPD to write an extension schema that declares the root element and the basic structure of the messages. The root element of the exchange brings together all of the objects and associations defined in the exchange. Whilst you are not required to create a separate schema to declare the root element and basic structure of the message, it can be beneficial to separate message-specific extensions into an 'exchange' schema and more generic extensions into 'extension' schemas. Exchange schemas contain definitions that are unique to a message type or group of message types. This generally includes only the root element and its type and possibly some structural elements that form the basic framework of the message.

	Organizing schema elements into 'exchange' and generic 'extension' groupings also offers the possibility of sharing the more generic schema across multiple IEPDs, whereas the 'exchange' schema is usually specific to one particular IEPD. You can also have multiple 'exchange' schemas in order to represent different message types or groups of different message types.

Subsetting NIEM with the Schema Composer

Enterprise Architect's Schema Composer is a tool that can greatly simplify the process of creating subsets from the NIEM Reference Model namespace Packages.

Access

Use either of the methods outlined here to display the Schema Composer window and then display the 'New Model Transform' dialog,

Enter a name for the new model transform, then from the 'Schema Set' drop-down list choose 'National Information Exchange Model (NIEM)'.

Save the profile as a Model Artifact within a suitable Package in your project (the root Package of your IEPD is suitable - then the Artifact will be easy to find).

Ribbon	Develop > Schema Modeling > Schema Composer > Open Schema Composer : New > Model Transform

Creating a Subset Model

NIEM experts suggest that a good first step is to create a UML model of your XML exchange, as it allows you to

capture your business requirements without being unduly influenced by how things are done in NIEM.

Once you have a first draft of a UML model for your exchange, you can then begin to re-create that model using NIEM.

Initially, finding appropriate types and properties within the NIEM Reference Model can seem to be an impossible task. This will become much easier as you gain experience and familiarity with the content of the NIEM model.

Most of the NIEM types that you will commonly use, such as PersonType, OrganizationType, DocumentType or ActivityType, have numerous attributes, of which you will generally require only a few. This is where subsetting becomes useful.

If you are trying to model a person, using their name, address and birthdate, you would choose PersonType and AddressType from NIEM-core. From those types, select only the properties that you require for your model.

Where the selected properties reference other types, those types will be automatically added to the Schema Composer.

When you 'generate' your subset, Enterprise Architect creates the target schema Packages required by the subset, then copies the selected types with their reduced attribute sets into the target Packages.

Further Refining Your Subset

Once you have created your subset, you can further refine it

by making adjustments to the cardinalities of the properties within the types or by restricting the allowed values of the properties.

To adjust the cardinality or restrict the permissible values of a property, select that property in the center pane of the Schema Composer, then right-click and choose 'Restrict this property'. The 'Property Restrictions' dialog is displayed, where you can adjust the cardinality or apply restrictions to the property as required.

Click on 'Update' to save the changes to your model transform profile, then click on 'Generate' to regenerate the subset model with the restrictions applied.

NIEM subsetting is often an iterative process. Using the saved model transform profile, you can reload, update and regenerate your subset as you require, throughout the various stages of IEPD development.

Subsetting NIEM Using the Schema Composer

Step	Action
1	Open the Schema Composer. (See *Access: Ribbon*)
2	Create a new Schema Composer profile.

	Click on the New button and select 'Model Transform'. In the dialog that opens, specify a name for the profile and select 'NIEM' in the 'Schema Set' field. (The 'Namespace' field on this dialog is not used for NIEM, as NIEM uses Tagged Values on its Model Packages to specify namespaces.) Choose a location to save your new profile, then click on the OK button.
3	In the Browser window, locate the required types PersonType and AddressType, in the NIEM-core Package of the Reference Model. Drag and drop the required types from the Browser window onto the 'Classes' pane of the Schema Composer.
4	Now select one of the types, say PersonType, in the Schema Composer's 'Classes' pane. The full list of attributes for PersonType is shown in the 'Attributes' pane.
5	Use the checkboxes in the 'Attributes' list to select the attributes of 'PersonType' to use in your exchange model. In this case, select the checkboxes for 'PersonBirthDate' and 'PersonName'. As you select these attributes, the Schema Composer automatically adds the types 'DateType' and

	'PersonNameType' to the list of Classes, as these types are referenced by the attributes you just selected.
6	Now select 'DateType' in the 'Classes' pane. 'DateType' has four attributes, DateAccuracyAbstract, DateAugmentationPoint, DateMarginOfErrorDuration and DateRepresentation. The first three of these attributes are date metadata - they do not hold a date value. The fourth, DateRepresentation, is an abstract attribute, so it does not directly hold date values either. It is used as a placeholder for the attribute that will ultimately hold the date value. The NIEM model commonly uses XML Schema abstract elements and substitution groups. The abstract elements add some complexity to the creation of a subset, because you are required to add the abstract element, as well as those elements that will be substituted in place of the abstract element. For example, most date-related types contain the abstract element nc:DateRepresentation that can be substituted by nc:Date, nc:DateTime, and so on.
7	Select the attribute DateType.DateRepresentation. You will notice that another type, DateRepresentationPropertyHolder has been added to the 'Classes' list.

8	Select DateRepresentationPropertyHolder in the 'Classes' list. The untyped attribute DateRepresentation is known as the 'head' of a substitution group. This attribute must be selected in the client of the substitution, DateType, as well as in the supplier of the substitution, DateRepresentationPropertyHolder. The attribute that is the head of the substitution group is pre-selected for you, so you only need to select the attribute that will eventually be substituted for DateRepresentation in DateType. Select the attribute Date:date - it will be used as the DateRepresentation that will actually hold a data value. Where substitution groups are involved, it is a common mistake to simply add the abstract element without also adding the substitutable element from the related PropertyHolder type.
9	Repeat the process for the PersonName attribute, by selecting PersonGivenName, PersonMiddleName and PersonSurName from the PersonNameType class.
10	To save your current selection of Classes and attributes to the profile you are creating, click on the Update button. This updates the profile with your current selection, allowing it to be reloaded at a later date if you need

	to perform further work on it. This facilitates an iterative process of creating the subset Package.
11	Now click on the 'Generate' option. Choose 'NIEM Model Subset' in the 'Schema Export' dialog and click on the Generate button. Navigate to the Package hierarchy containing the IEPD that you are building. Select the parent Package that will contain the subset Packages, then click on the OK button.
12	The Classes you have selected in the Schema Composer will be copied to the target Packages, with just the subset of attributes that you have selected.

Notes

- Please read through each of the walk-through examples - each one contains important information
- The Schema Composer functionality that supports NIEM development, assists in creating Subset Schemas; it does not assist in producing Extension Schemas

Walk Through Examples

If you are new to using the Schema Composer for NIEM, please take the time to read through these examples. Each example contains important information that will help to ensure that your models use valid NIEM subsets, which will ultimately produce valid XML schema files.

Example 1: Adding Classes and Selecting Attributes

This 'walk-through' example demonstrates how to use Enterprise Architect's Schema Composer to perform basic operations of adding Classes and selecting attributes to be included in a NIEM subset Package.

Step	Description
1	Open an Enterprise Architect project that contains the NIEM 5.0 Reference Model and also the NIEM IEPD Types. If you don't have such a project, then open a new project and load the Reference Model and IEPD Types, using the Model Builder (Ctrl + Shift + M).
2	Using the Model Builder, add a fresh copy of the NIEM 5 IEPD Starter Model to your project. You should rename the object instance 'NIEM-IEPD'

	to something more meaningful. When generating the IEPD, the name of this object instance is used to name the root folder in which the IEDP is created. If you wish, rename the Package 'NIEM 5 Starter Model' to something more appropriate as well.
3	The starter model contains a Schema Composer artifact named 'Schema Composer profile - NIEM 5 subset'. Locate this artifact in the Browser window, then double-click on it. This will open the Schema Composer and load the profile 'Schema Composer profile - NIEM 5 subset'. The lower part of the Schema Composer contains three columns. From left to right, they are labelled 'Classes', 'Attributes' and 'Schema'.
4	Using the Browser window, locate the Package 'niem-core' in the NIEM 5.0 Reference Model. Within that Package, locate the Class 'AircraftType'. Drag and drop 'AircraftType' onto the left-hand column of the Schema Composer (labelled 'Classes'). You will notice that Classes 'ConveyanceType' and 'ItemType' are added automatically to the list of Classes. 'ItemType' and 'ConveyanceType' are supertypes from which 'AircraftType' is derived.
5	Select AircraftType in the 'Classes' column.

	You will notice that the center column, 'Attributes', displays the full list of attributes belonging to this Class. The attributes of the parent Classes are also listed. To include an attribute in the subset schema, you simply place a checkmark beside it. (You should choose only attributes of the Class that is currently selected in the 'Classes' list. If you require attributes of a parent Class, select that Class, then select its attributes.) Place a checkmark beside AircraftTailIdentification. The type of AircraftTailIdentification is IdentificationType. Notice that IdentificationType has been added to the list of Classes. Enterprise Architect automatically adds to the 'Classes' list, those classifiers that are referenced as types of the attributes you select.
6	Select the Class IdentificationType in the left-hand column of the Schema Composer. In the center column, place a check mark beside the attribute IdentificationID. The type of IdentificationID is 'string'. The type 'string' is a Primitive type - it is not added to the list of Classes.
7	Now, select the Class ConveyanceType in the left-hand column of the Schema Composer.

	Place a check mark beside the attribute ConveyanceMotorizedIndicator. The type 'boolean' is a Primitive type - it is not added to the list of Classes.
8	Select the Class ItemType in the left-hand column of the Schema Composer. Place check marks beside the attributes ItemMakeName, ItemModelName and ItemModelYearDate. The types ProperNameTextType and TextType are automatically added to the list of Classes. TextType is the base Class for ProperNameTextType.
9	Click on the Update button to save the selected Classes and attributes to the profile, then click on the Generate button. In the window that opens, select 'NIEM Model Subset', then click on the Generate button. You will be prompted to select a Package within which the subset model will be created. Typically, you would choose the Package that is the parent of the Exchange schema Package. In the starter model, the exchange Package is named 'IEPD-Exchange' and its parent Package is named 'NIEM 5 Starter Model', although you might have renamed these earlier in step 2. Select the Package 'NIEM 5 Starter Model', then

	click on the OK button. Note: When creating more complex models, your subset might include Classes from several different <<InformationModel>> Packages. Enterprise Architect's Schema Composer will automatically create the required target Packages and copy the Classes you are subsetting into the target Packages whose Tagged Value 'targetNamespace' matches that of the source Package from which the original Class was drawn. The subset <<InformationModel>> Packages will be created as children of the Package you choose as the generation target.
10	Once the generation is complete, expand the target <<InformationModel>> Packages. You will see the Classes you selected with their reduced sets of attributes.

Example 2: Using Association Types

This 'walk-through' example demonstrates how to use Enterprise Architect's Schema Composer to add Association Types and the types they reference, to your NIEM subset Package.

Step	Description

1	Open an Enterprise Architect project that contains the NIEM 5.0 Reference Model and also the NIEM IEPD Types. If you don't have such a project, then open a new project and load the Reference Model and IEPD Types using the Model Builder (Ctrl + Shift + M).
2	Using the Model Builder add a fresh copy of the NIEM 5 IEPD Starter Model to your project. You should rename the object instance 'NIEM-IEPD' to something more meaningful. When generating the IEPD, the name of this object instance is used to name the root folder in which the IEDP is created. If you wish, rename the Package 'NIEM 5 Starter Model' to something more appropriate as well.
3	The starter model contains a Schema Composer artifact named 'Schema Composer profile - NIEM 5 subset'. Locate this artifact in the Browser window, then double-click on it. This will open the Schema Composer and load the profile 'Schema Composer profile - NIEM 5 subset'. The lower part of the Schema Composer contains three columns. From left to right, they are labeled 'Classes', 'Attributes' and 'Schema'.
4	Using the Browser window, locate the Package 'niem-core' in the NIEM 5.0 Reference Model. Within that Package, locate the Class

	'PersonLocationAssociationType'. Drag and drop 'PersonLocationAssociationType' onto the left-hand column of the Schema Composer (labelled 'Classes'). You will notice that the center column 'Attributes' displays PersonLocationAssociationType.Attributes and also PersonLocationAssociationType.Associations. Place check marks beside both of the associations, Location and Person. The types LocationType and PersonType are automatically added to the Schema Composer's 'Classes' list.
5	The Class PersonLocationAssociationType is derived from the supertype 'nc:AssocationType', but in this case the supertype is not automatically added to the Classes list. If you want to include any attributes of the supertype 'nc:AssociationType' in your generated subset, you must add 'nc:AssociationType' to the Schema Composer's Class list manually, then select the required attributes. If you don't want to specifically include attributes of 'nc:AssociationType', then there is no need to add it to the Classes list. When the schema file is eventually generated from the subset Package, Enterprise Architect will

	generate an element and type definition for 'nc:AssociationType' if and when it is required, even if it is not explicitly modeled.
6	Click on the Update button, then click on the Generate button. In the window that opens, select 'NIEM Model Subset', then click on the Generate button. You will be prompted to select a Package within which the subset model will be created. Typically, you would choose the Package that is the parent of the Exchange schema Package. In the starter model, the exchange Package is named 'IEPD-Exchange' and its parent Package is named 'NIEM 5 Starter Model', although you might have renamed these earlier in step 2. Select the Package 'NIEM 5 Starter Model', then click on the OK button.
7	Locate the <<InformationModel>> Package named 'niem-core' within the subset model. Create a NIEM PIM diagram within this Package, then drag and drop the three Classes in this Package onto the diagram. You will notice that the properties 'Person' and 'Location' are modeled as AssociationEnds on the Associations between PersonLocationAssociationType and the types PersonType and LocationType.

Example 3: Using Substitution Groups and Property Holders

This 'walk-through' example demonstrates how to use Enterprise Architect's Schema Composer to correctly add substitution groups and property holders to your NIEM subset Package.

Step	Description
1	Open an Enterprise Architect project that contains the NIEM 5.0 Reference Model and also the NIEM IEPD Types. If you don't have such a project, then open a new project and load the Reference Model and IEPD Types using the Model Builder.
2	Using the Model Builder, add a fresh copy of the NIEM 5 IEPD Starter Model to your project. You should rename the object instance 'NIEM-IEPD' to something more meaningful. When generating the IEPD, the name of this object instance is used to name the root folder in which the IEDP is created. If you wish, rename the Package 'NIEM 5 Starter Model' to something more appropriate as well.
3	The starter model contains a Schema Composer

	artifact named 'Schema Composer profile - NIEM 5 subset'. Locate this artifact in the Browser window, then double-click on it. This will open the Schema Composer and load the profile 'Schema Composer profile - NIEM 5 subset'. The lower part of the Schema Composer contains three columns. From left to right, they are labeled 'Classes', 'Attributes' and 'Schema'.
4	Using the Browser window, locate the Package 'niem-core' in the NIEM 5.0 Reference Model. Within that Package, locate the Class 'AircraftType'. Drag and drop 'AircraftType' onto the left-hand column of the Schema Composer (labeled 'Classes'). You will notice that the Classes ConveyanceType and ItemType are added automatically to the list of Classes. ItemType and ConveyanceType are supertypes from which AircraftType is derived.
5	Select the Class 'AircraftType' in the left-hand column of the Schema Composer. In the center column, place a check mark beside the attribute AircraftWingColorAbstract (notice that this attribute has no type specified). The Class AircraftWingColorAbstractPropertyHolder is automatically added to the list of Classes.

6	Select the Class 'AircraftWingColorAbstractPropertyHolder' in the left-hand column. Notice that this Class also has an attribute named 'AircraftWingColorAbstract' that has no type specified. This attribute is pre-selected for you - it should remain selected. Simply place a check mark beside AircraftWingColorText.
7	In this case, the attribute AircraftWingColorAbstract is the head of the substitution group and provides the connection between the client Class AircraftType and the supplier Class AircraftWingColorAbstractPropertyHolder. AircraftWingColorText is the actual attribute (of type TextType) that will be added to AircraftType.
8	Some PropertyHolder types will have several attributes - the substitution group head, plus a number of others. The attribute that is the head of the substitution group must always be selected in both the client and the supplier Classes. Enterprise Architect pre-selects this attribute for you in the supplier Class (the PropertyHolder). You then only need to select the attribute(s) from the supplier that you want to substitute in place of the head of the substitution group.

Example NIEM Schema

This page provides an overview of defining a new NIEM compliant schema, from start to finish.

The framework Packages that are required for NIEM modeling have been described in previous topics. The Model Builder also provides a Package that acts as a convenient starting point for defining your IEPD. When this is imported to your model you will find diagrams containing instances of the IEPD types, with the run-state set to show the core properties that you are most likely to need to set.

This section describes the process of taking the sample IEPD from the Pattern, and creating a 'Hello World' style message, where a request is made for a personalized message based on a facial image. The response will be the identity of the pictured person and a personalized message for them.

Import NIEM framework Packages

Modeling with NIEM in Enterprise Architect starts with the standard types defined by the NIEM Technical Architecture Committee and the Object Management Group NIEM-UML specification, as described here. These are available from our Reusable Asset Server and the Model Builder.

To import these into your model:

- Open the Model Builder (Ctrl+Shift+M)
- In the Model Builder dialog, click on the <perspective

name> button, then select the perspective 'Information Exchange > NIEM'

- Expand the item 'NIEM 3, 4 and 5'
- Select the Packages required for your model
- Click on the Create Model button to import the selected patterns into your model

Note:

- All NIEM 5 models require the NIEM IEPD Types Package as well as one of the NIEM Reference Model Packages
- All NIEM 3 and 4 models require the NIEM MPD Types Package as well as one of the NIEM Reference Model Packages
- All NIEM 2.1 models require the NIEM 2.1 Reference Model Package but not an MPD Types Package, as the NIEM 2.1 MPD elements are available from the NIEM 2.1 MPD Diagram Toolbox

Component	Details
NIEM Framework	The power of NIEM comes primarily from the extensive library of types that you can use to build your own schemas. Enterprise Architect provides complete NIEM frameworks for NIEM 5, as well as all versions of NIEM 3 and NIEM 4. These frameworks are all available from the Model Builder. This tutorial is using the NIEM 5.0

	framework, so select that pattern for import.
IEPD Types from NIEM-UML	A user-defined NIEM schema is built around an IEPD that defines, for consumers of the schema, how to use the various XSD files that are included and what message types are being defined. When modeling in UML, an IEPD is created using instances of a number of Classes defined in the UML Profile. Enterprise Architect provides these Classes in a Package that is available from the Model Builder. All NIEM 5 models will require these IEPD types, so select the 'NIEM 5 IEPD Types' pattern for import.

Subset NIEM Namespaces

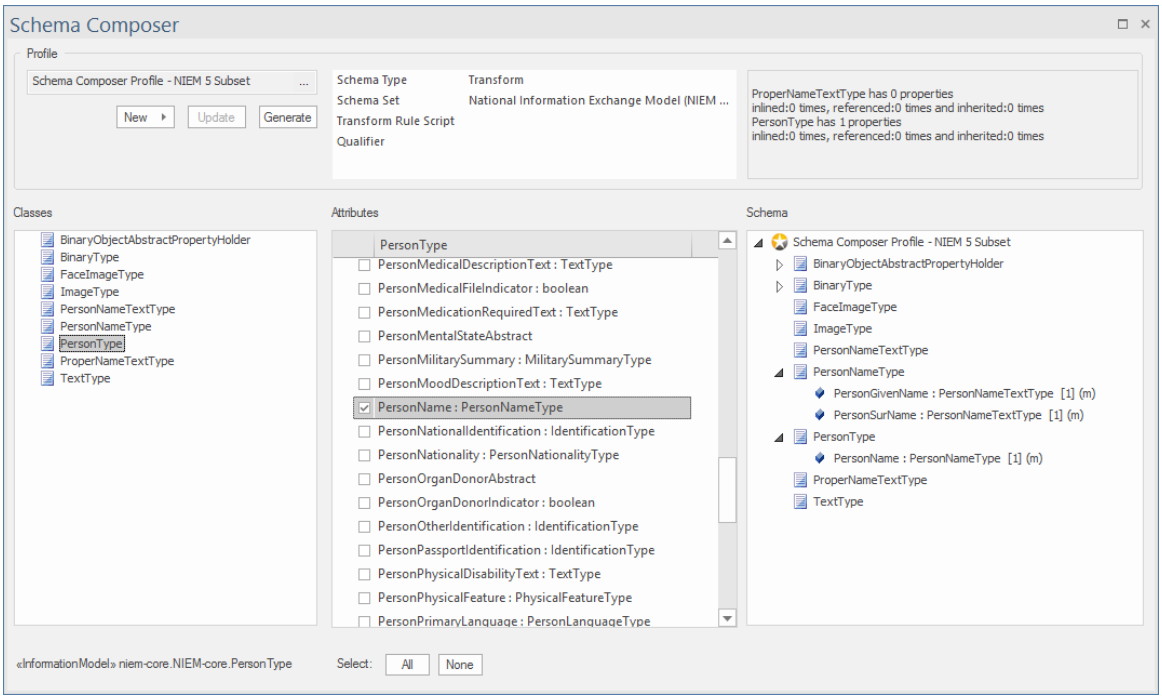
The Starter Model pattern includes a Schema Composer Artifact to use for specifying a subset. Double-click on it to open the Schema Composer and begin the subsetting process.

We want our request message to send a facial image to be used for facial recognition. To do that, we need to subset the appropriate types from the Biometrics Package. Start by locating the type FaceImageType in the Domains\Biom

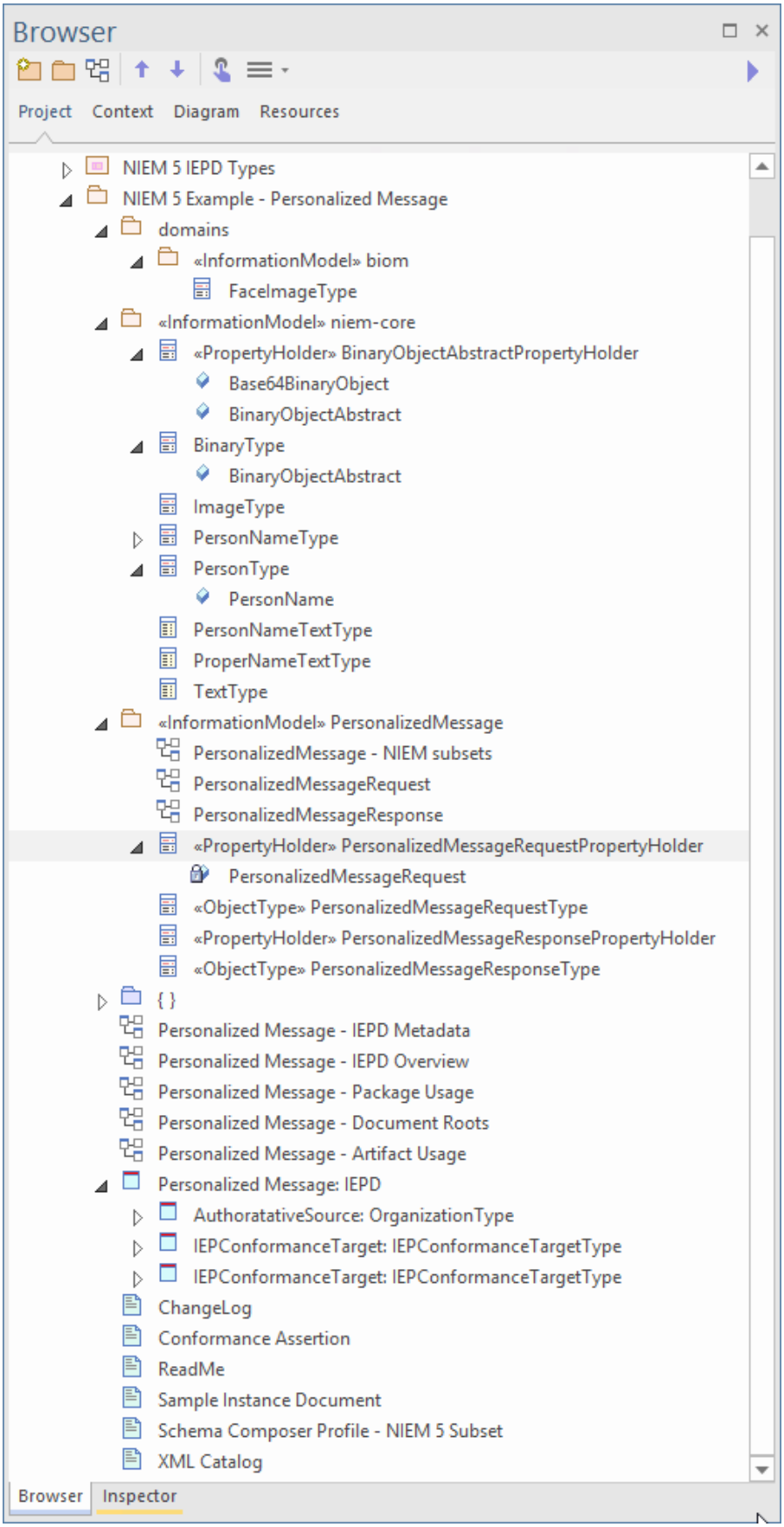
Package within the NIEM 5.0 Reference Model. Drag this type into the Schema Composer. The super-types from which this type inherits are automatically added to the Schema Composer.

Our response message requires PersonType from the 'niem-core' Package. Drag this type onto the Schema Composer as well.

This image shows the selection of a subset of the types and properties across a number of namespaces within the NIEM 5.0 Reference Model:

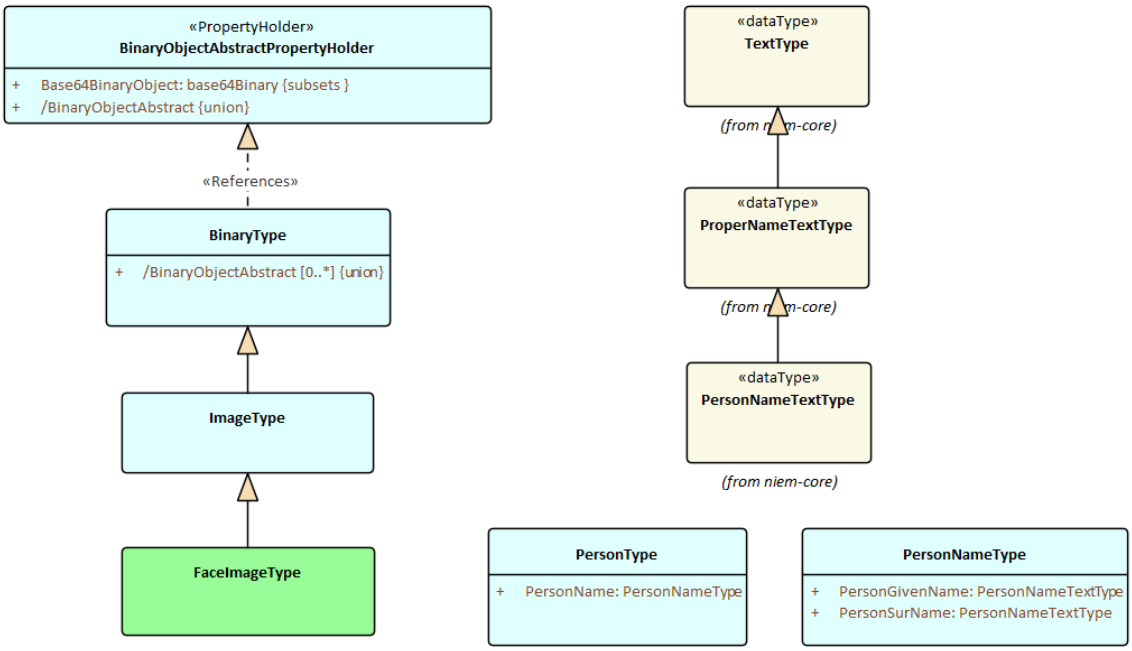


Once the required types are selected, you can generate the subset. When prompted, select the **parent** Package into which the subset namespace Packages will be generated. After generation, the Classes in the subset Packages should resemble this:



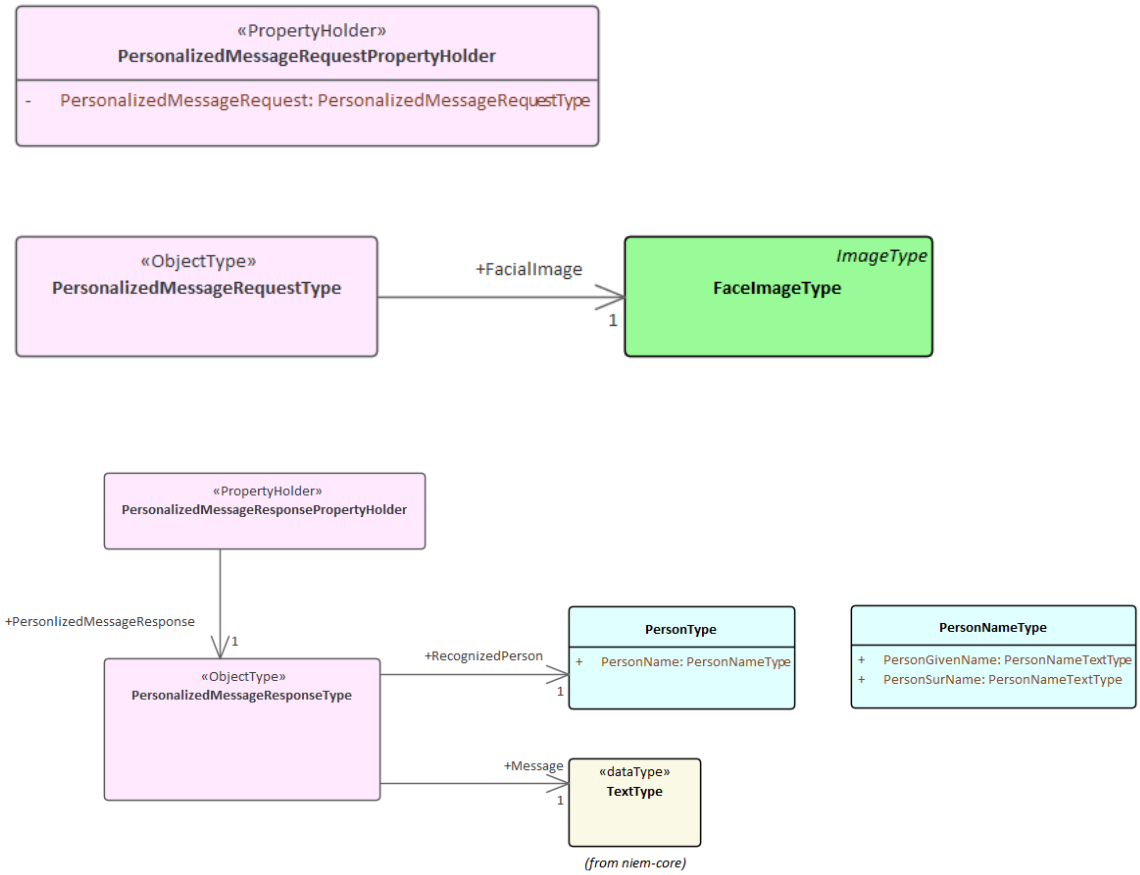
We can now create a NIEM PIM diagram and place all our

subset Classes on that diagram, to produce something that resembles this:



Create extension types

We will be defining two messages, a request and a response. For each of these messages, we need to define the document root elements. These will be modeled as extensions to the NIEM schema. Now that we have defined our subset Packages we can define these document roots. Because we are only creating two simple document types, all that is needed is a PropertyHolder and ObjectType for each message. The ObjectTypes link to the types we've selected from the NIEM framework, to describe the contents of each message as shown:



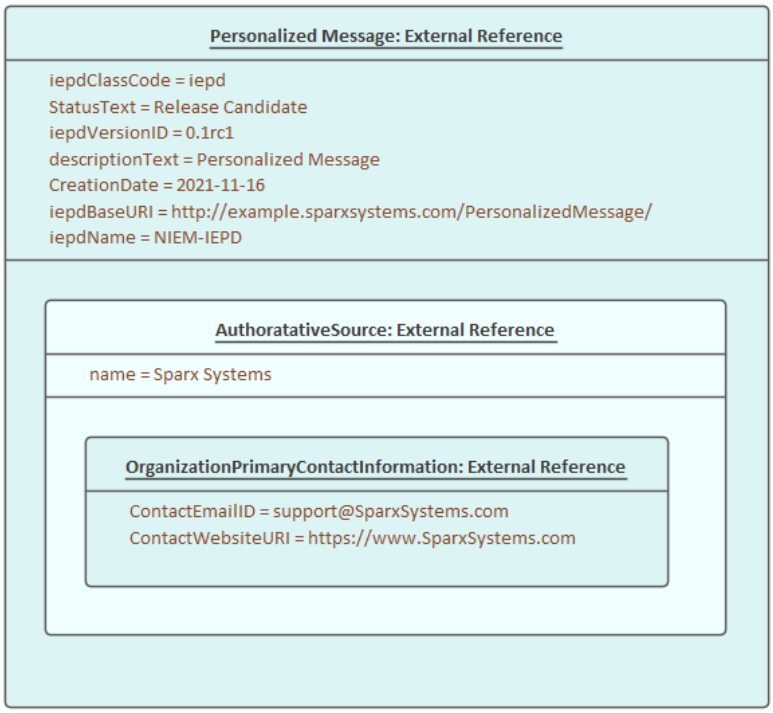
Customize the IEPD

The instance of the IEPD Class carries information that identifies the IEPD, and the links between it and the various other model artifacts determine what is generated (and where it is generated to) when generating the schema files and catalog files. The main points are described here.

Component	Description
IEPD Metadata	The top level object in the Pattern is an instance of the IEPD class. The name of the IEPD is the name of the Object itself. All other properties are in the Run-State

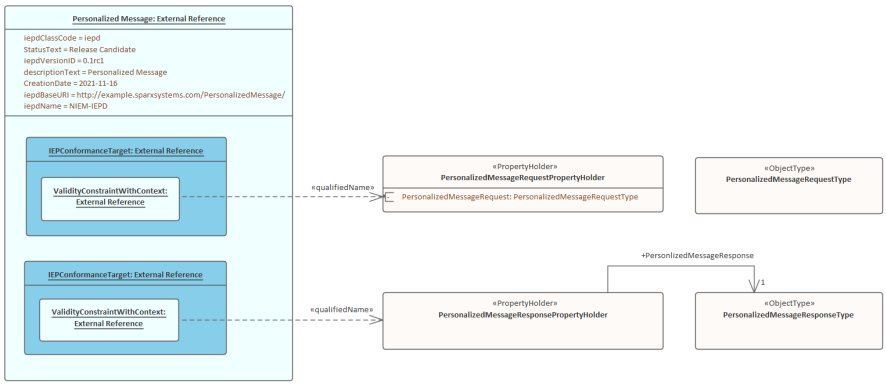
of the object.

This figure shows how the IEPD might look after providing real information.



NIEM-UML recommends the last section of iepdBaseURI matches the name of the IEPD, and specifies that the iepdVersionID will be appended to the iepdBaseURI to produce the generated iepdURI. This example follows that convention.

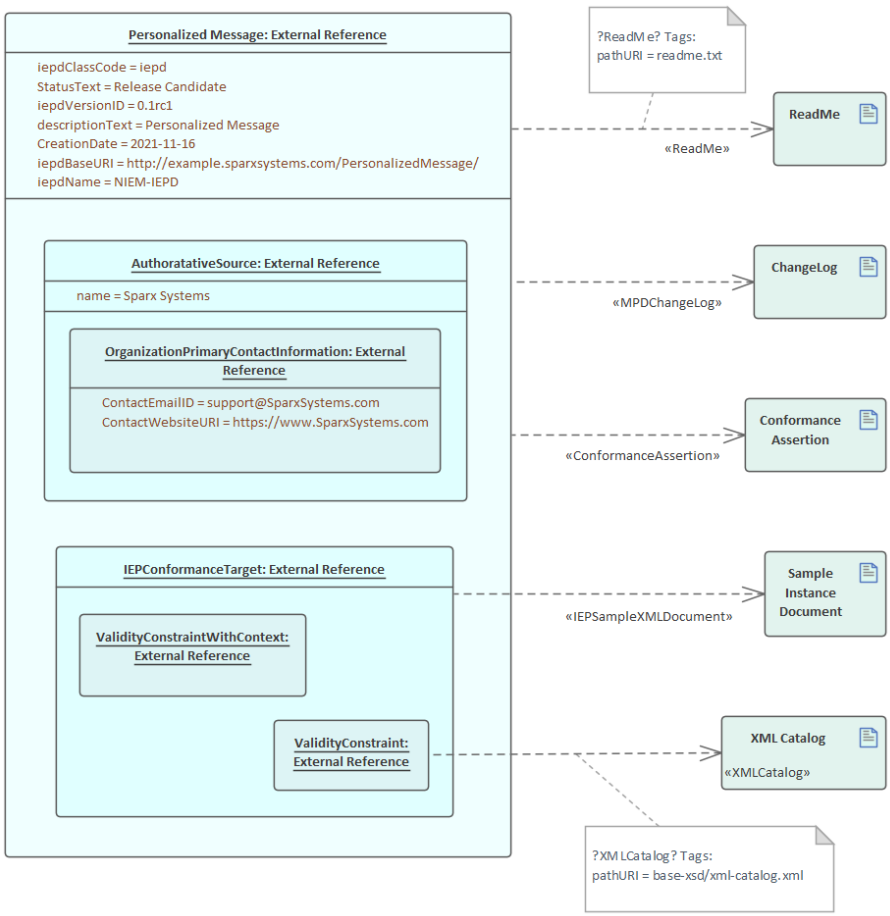
The Pattern defaults the value of iepdClassCode to 'iepd'. This means that the IEPD is intended to represent an Information Exchange Package Document (IEPD). This is the most common type of IEPD, and it is what we want to create, so it has been left with the

	default value.
Defined Document Types	An IEPD is expected to define one or more document types. Each one will be an instance of IEPConformanceTargetType named 'IEPConformanceTarget'. The provided model Pattern already includes one of these, but we need a second one as shown here:  Note the instances of QualifiedNamesType, with the qualifiedName relationship to a PropertyHolder. This specifies that the top level of the document being described will be an element from one of the contained attributes. The section *Create Extension Packages* in the topic *Creating a NIEM Data Model* describes how this is defined.
Package Usage	The relationships connecting the IEPD instance to the Information Models

	specify which schema files are to be generated with this IEPD. In this example we use types from two different NIEM namespaces. The sub-setting process has created an InformationModel Package for each, where the Namespace Tagged Values match the original, and the purpose is set to subset. We also create an extension Package where we define our own types and how the NIEM types will be used. This figure shows how this looks: The relationships used also specify how the Package is used and the relative path to the schema defined by that Package.
Additional Files	All IEPD Packages are expected by NIEM to contain - at a minimum - a change log and a readme, but there are

several other types of Artifact that are also supported. In Enterprise Architect, each is defined using a stereotyped relationship to an Artifact. As with the Package use, the relationship specifies where the file will be located.

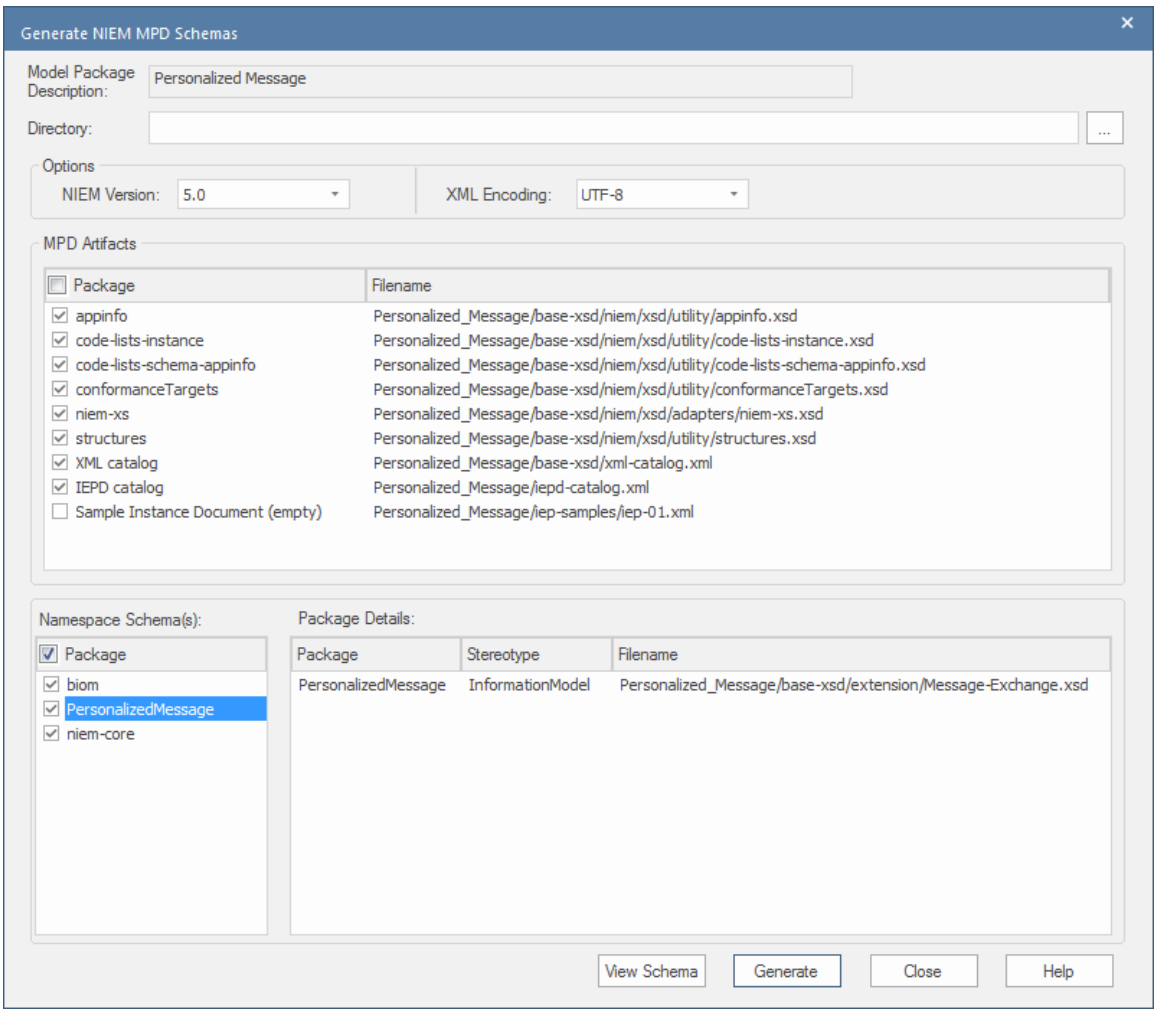
In this image a ReadMe, ChangeLog and a sample document are described for each of the document types. This will add information about those files to the target catalog file. The files will not be created by Enterprise Architect, and their content is beyond the scope of this tutorial.



Generate IEPD

To generate your IEPD:

- Select the IEPD instance specification, either on the diagram or in the Browser window.
- From the 'Specialize' ribbon, select the option 'Technologies > NIEM > Generate NIEM Schema'



The dialog shows the standard NIEM artifacts and the list of linked namespaces that can be generated as schemas. Set the target directory and click on the Generate button to create the modeled IEPD.

Import NIEM XML Schema

As well as generating NIEM schema in Enterprise Architect, you can import (reverse engineer) an external NIEM-specific XML Schema file into your Enterprise Architect project as a UML model.

Access

Ribbon	Specialize > Technologies > NIEM > Import NIEM Schema Specialize > Technologies > NIEM 2.1 > Import NIEM 2.1 Schema
Context Menu	In the Browser window: Right-click Package | Specialize | NIEM | Import NIEM Schema Right-click Package | Specialize | NIEM 2.1 | Import NIEM 2.1 Schema

Import a NIEM specific XML Schema

Option	Action
Package	Displays the name of the

	currently-selected Package in the Browser window, as the Package into which to import the NIEM schema. You can verify that you are using the appropriate Package by clicking on the button and checking the 'Navigator' dialog; select a different Package if necessary.
Directory	Click on the button and browse for the directory containing the source NIEM Schema file(s). Click on each file to import, and then click on the browser Open button.
Selected File(s)	Lists the XML Schema file(s) selected for import.
Import referenced XML Schema(s)	Select this checkbox if you want to import any other XML Schema that is referenced by any of the files listed in the 'Selected File(s)' field.
Skip Schema if Namespace in Model	Select this checkbox if you want to skip importing an XML Schema if it already exists in the model. Enterprise Architect will use the Schema namespace and name to determine if it exists in the Model.

Create Diagram for XML Schema(s)	Select this checkbox to create a Class diagram (a NIEM PIM diagram) under each imported Namespace Package.
Layout created Diagram	(Enabled only if the 'Create Diagram for XML Schema(s)' option is selected.) Select this checkbox to automatically lay out the created Class diagram(s).
Import	Click on this button to start the import process. The progress of the import is reported in the 'NIEM Importer' tab of the System Output window. A message box also displays to indicate when the import is complete; click on the OK button to clear the message.
Close	Click on this button to close the 'Schema Importer' dialog.
Help	Click on this button to display this Help page.

Notes

- Enterprise Architect uses the *schemaLocation* attribute in the XSD Import and XSD Include elements of an XML Schema, to determine the dependencies between the files; this attribute must be set to a valid file path (and not a URL) for the dependent XML Schema(s) to be imported correctly
- The 'Create Diagram for XML Schema(s)' option generates a diagram for each imported schema file, but displays the diagrams only for the schema files specifically selected by the user; it does not display the diagram for a referenced schema file
- If you import large schema files, it is recommended that you deselect the 'Create Diagram for XML Schema(s)' option, as this considerably increases the time taken by the import