European Network of
Transmission System Operators
for Electricity

CGMES Profiling User Guide v1.0

2021-04-21

SOC APPROVED
VERSION 1.0

Maintenance notice:

This document is maintained by the ENTSO-E CIM EG. Comments or remarks are to be provided at cim@entsoe.eu

NOTE CONCERNING WORDING USED IN THIS DOCUMENT

The force of the following words is modified by the requirement level of the document in which they are used.

- SHALL: This word, or the terms “REQUIRED” or “MUST”, means that the definition is an absolute requirement of the specification.
- SHALL NOT: This phrase, or the phrase “MUST NOT”, means that the definition is an absolute prohibition of the specification.
- SHOULD: This word, or the adjective “RECOMMENDED”, means that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- SHOULD NOT: This phrase, or the phrase “NOT RECOMMENDED”, means that there may exist valid reasons in particular circumstances when the particular behaviour is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behaviour described with this label.
- MAY: This word, or the adjective “OPTIONAL”, means that an item is truly optional.

Revision History

Version: Date	***Author***	***Comment***
1.0: 21-April-2021	*Chavdar Ivanov*	*Version for first drafted by CGMES SG, agreed in CIM EG, for SOC approval.* *Approved by SOC.*

CONTENTS

List of figures

INTRODUCTION

This document describes the way to develop and maintain CIM profiles using CimConteXtor, CimSyntaxGen and Enterprise Architect. It includes a set of modelling rules that are defined for the CIM Model Manager activities as they are relevant to the profiling work and not published elsewhere. When the rules are standardised the user guide shall be revised to only refer to these rules. The document covers the needs of profiling in the frame of IEC 61970-400 series and IEC 61970-600 series of standards, but the techniques can be applied for other profiles too.

Scope

The objective of this guide is to enable a person to work based on the common information model (CIM) which currently includes IEC 61968-11, IEC 61970-301 (which also includes European CIM extensions), IEC 61970-302 and IEC 62325-301 and in particular:

- to modify profiles published in the IEC 61970-400 series and IEC 61970-600 series standards;
- to develop new profile related standards within the IEC 61970 series for international standardization purpose.

The prerequisites for this guide are as follows:

- The person shall have some knowledge about UML modelling.
- The person shall have the Enterprise Architect version 14 or later from Sparx. Using earlier versions of Enterprise Architect might be possible but inconsistencies maybe observed.
- The person shall have some knowledge about XML (RDF).
- The person shall have the Add-Ins CimConteXtor (version 2.9.24 or later) and CimSyntaxGen (version 3.5.13 or later) installed. These add-ins are available here: www.cimcontextor.net.
- The person shall have the latest version of the jCleanCim to validate the model and generate the documentation. jCleanCim is available here: http://www.tanjakostic.org/jcleancim/.
- The person shall have the latest merged CIM model which is normally available in the CIMug SharePoint and in the ENTSO-E SharePoint.
- The person shall be acquainted with the CIM, either by buying the IEC international standards or using the draft versions from the CIMug SharePoint or from elsewhere.
- The person shall be acquainted with the ENTSO-E business processes for the market.

Note: all the information and files are available in the ENTSO-E SharePoint and CIMug SharePoint. The ENTSO-E SharePoint is only for ENTSO-E members. IEC international standards are copyrighted protected and thus they are only available from IEC webstore site, CEN/CENELEC site or National standardization organization.

Parts of the document are depending on:

- IEC 61970-401 and CIM Model Management document (CIMug document). These documents are drafts. The current version of CIM Model Management document is published here: https://cimug.ucaiug.org/Model%20Manager%20Documents/Public/CIM%20Modeling%20Guide_v1.1.pdf
- Sparx Enterprice Architect and CimConteXtor, CimSyntaxGen user manual

The following figure illustrates the dependency between this document and other documents.

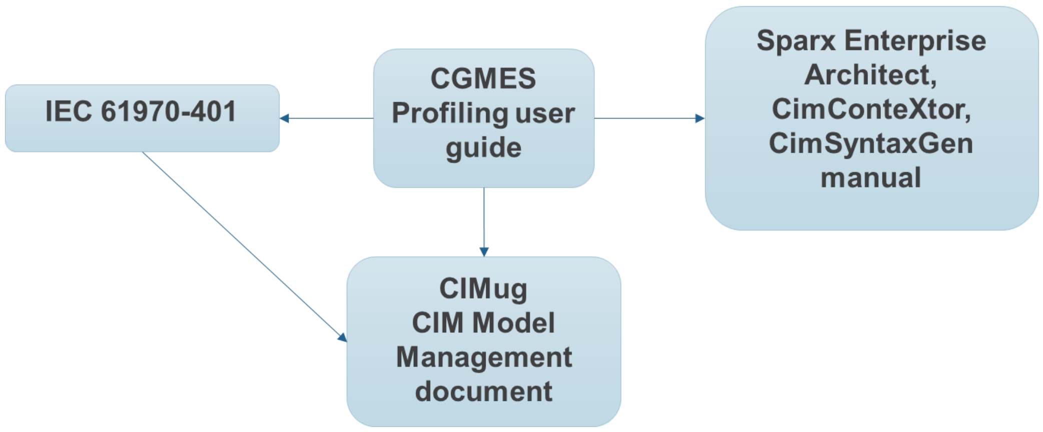

IEC 61970-401
CGMES Profiling user guide
Sparx Enterprise Architect, CimConteXtor, CimSyntaxGen manual
CIMug CIM Model Management document

Figure 1 – Profiling user guide document dependencies

Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 61970-401, E*nergy management system application program interface (EMS-API) - Part 401: Profile framework*

Terms and definitions

For the purposes of this document, the terms and definitions given in IEC 61970-2, as well as the following apply.

3.1
Common Information Model
CIM
abstract model that represents all the major objects in an electric utility enterprise typically contained in an EMS information model

Note 1 to entry: By providing a standard way of representing power system resources as object classes and attributes, along with their relationships, the CIM facilitates the integration of EMS applications developed independently by different vendors, between entire EMS systems developed independently, or between an EMS system and other systems concerned with different aspects of power system operations, such as generation or distribution management.

[SOURCE: IEC 61970-552:2016]

3.2
Profile
schema that defines the structure and semantics of a model that may be exchanged

Note 1 to entry: A Profile is a restricted subset of the more general CIM.

[SOURCE: IEC 61970-552:2016]

3.3
Profile Document
collection of profiles intended to be used together for a particular business purpose

[SOURCE: IEC 61970-552:2016]

3.4
Resource Description Framework
RDF
language recommended by the W3C for expressing metadata that machines can process simply

Note 1 to entry: RDF uses XML as its encoding syntax.

[SOURCE: IEC 61970-552:2016]

3.5
RDF Schema

schema specification language expressed using RDF to describe resources and their properties, including how resources are related to other resources, which is used to specify an application-specific schema

[SOURCE: IEC 61970-552:2016]

3.6
Unified Modelling Language
UML
object-oriented modelling language and methodology for specifying, visualizing, constructing, and documenting the artefacts of a system-intensive process

[SOURCE: IEC 61970-552:2016]

Basic concepts

4.1 Overview of the modelling methodology

In general, the IEC 61970-401[1] document specifies the structure of a profile specification and the rules for creating the subsets from the canonical CIM. Profile documents describe a subset of the canonical CIM dedicated to a specific data exchange, the canonical CIM is described in the IEC 61970-300 series documents as well as 61968-11. The guiding principle for the profiling method is that the information described by a profile is a true subset of the canonical CIM and retain class, role and attribute names from the canonical CIM. The main objective of it is that different datasets exchanged using different profiles based on canonical CIM solely rely on the definitions and basic principles of the canonical CIM which is a key to make interoperability efforts feasible. This also enables different profiles to relate data between them by using the canonical CIM as a hub and supports a reader of a data set or a message to easily find descriptions of elements in both the profile and the canonical CIM.

There are several languages that can describe profiles, e.g. UML or OWL. UML includes a graphical language that is implemented by UML editors. OWL does not have a graphical language, but several editors exist that support the display and editing of OWL data. The languages in which profiles can be described are outside the scope of this specification as well as how profiles are presented and edited in user interfaces.

A profile in UML is described by classes, attributes, associations and roles, the common way to describe information in UML. Hence profiling with UML may mean copying and updating classes and associations from the canonical CIM. A profile in OWL is described by classes and properties. There are two types of OWL properties matching with UML attributes and UML roles. Profiling in OWL means creating OWL classes and properties by selecting UML classes, attributes and roles from canonical CIM the same way as it is done for profiling with UML.

This user guide explains profiling techniques that describe data exchanged with CIMXML files according to IEC 61970-552. Tools that process data described by profiles will need a machine-readable version of the profiles. IEC 61970-501 is an RDF based serialization intended for this.

4.2 Canonical CIM UML

There are two kinds of Enterprise Architect file that can be used:

- The CIM file from the IEC CIM model manager: This file is the result of the merge of the three CIM packages, i.e. IEC61968, IEC61970 and IEC62325. The name of the file is as follows: iec61970cimNNvNN_iec61968cimNNvNN_iec62325cimNNvNN.eap, where NN is a two digit value providing for release and version of each package. As an example, iec61970cim17v38_iec61968cim13v12_iec62325cim03v17a.eap is the CIM file with version 17v38 of IEC61970 package, version 13v12 of IEC61968 package and version 03v17 of IEC62325 package (see Figure 2).

[1] It is in a CDV stage with deadline January 2020.

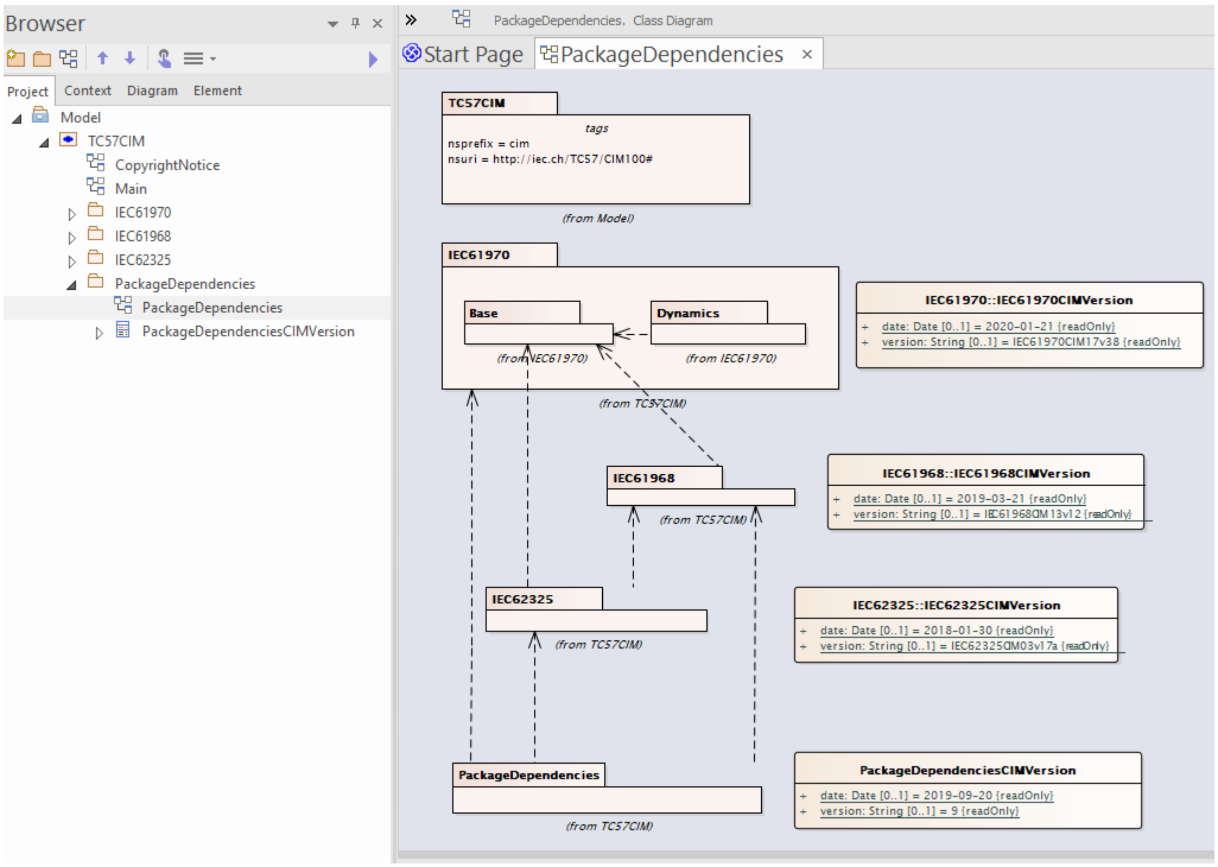

Browser
Project
Context
Diagram
Element
Model
TC57CIM
CopyrightNotice
Main
IEC61970
IEC61968
IEC62325
PackageDependencies
PackageDependencies
PackageDependenciesCIMVersion
PackageDependencies. Class Diagram
Start Page
PackageDependencies
TC57CIM
tags
nsprefix = cim
nsuri = http://iec.ch/TC57/CIM100#
(from Model)
IEC61970
Base
(from IEC61970)
Dynamics
(from IEC61970)
(from TC57CIM)
IEC61970::IEC61970CIMVersion
+ date: Date [0..1] = 2020-01-21 {readOnly}
+ version: String [0..1] = IEC61970CIM17v38 {readOnly}
IEC61968
(from TC57CIM)
IEC61968::IEC61968CIMVersion
+ date: Date [0..1] = 2019-03-21 {readOnly}
+ version: String [0..1] = IEC61968CIM13v12 {readOnly}
IEC62325
(from TC57CIM)
IEC62325::IEC62325CIMVersion
+ date: Date [0..1] = 2018-01-30 {readOnly}
+ version: String [0..1] = IEC62325CIM03v17a {readOnly}
PackageDependencies
(from TC57CIM)
PackageDependenciesCIMVersion
+ date: Date [0..1] = 2019-09-20 {readOnly}
+ version: String [0..1] = 9 {readOnly}

Figure 2 – Enterprise Architect file containing canonical CIM only

- The CIM file which contains profiles: This file is the result of the merge of the file with canonical CIM and the profiles. In practice there might be multiple such files, but this is discussed in section 4.4. For instance, for CGMES profiles the name of the EA project file can be FDIS06-iec61970-600-CGMES-v3_0_0.eap which indicates the related IEC standard and its standardisation stage.

4.3 UML package structure

Figure 3 describes structure of an Enterprise Architect file which contains profiles.

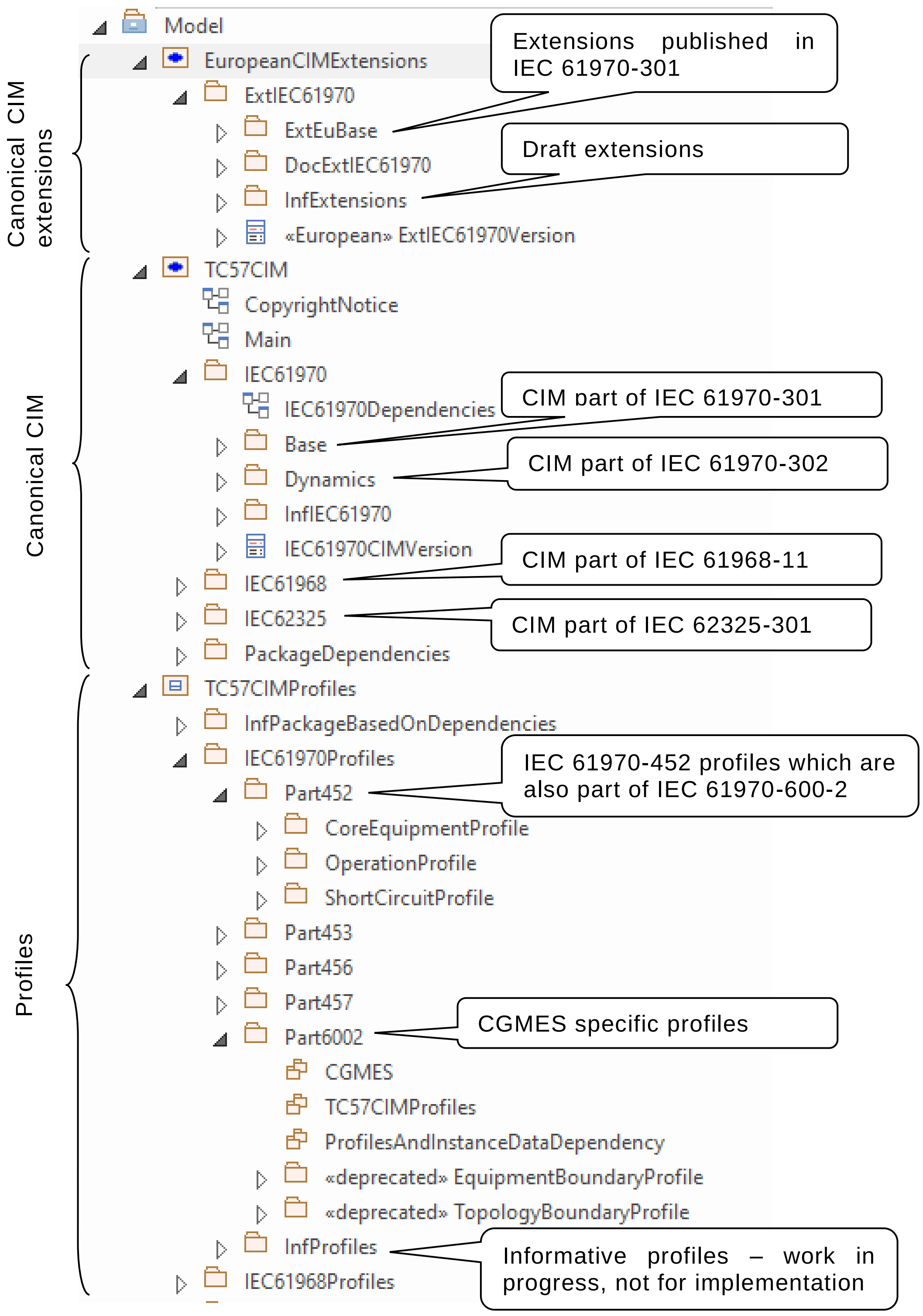


Figure 3 – Overview of EA file which contains profiles

It is important to understand that the structure of the EA file is made as such to facilitate the maintenance of the UML model and profiles (see also section 4.4). As the example in Figure 3 contains profiles it is obvious to see three main packages:

1) “EuropeanCIMExtensions”, which contains the extensions done on top of the canonical CIM. Please note that such extensions are also part of the canonical model and are published in the respective IEC standards describing canonical CIM;

2) “TC57CIM” which contains the canonical CIM, and

3) “TC57CIMProfiles” which contains the profiles developed in the scope of TC57. In case these profiles contain specific extensions applicable only for a certain IEC standard there is a mechanism to filter out what is not necessary at the time of the document generation. The mechanism is using stereotypes to identify different extensions and use filtering approach of jCleanCim.

As the profiles are currently developed using CimConteXtor the dependencies between the profiles and the canonical CIM packages is essential. Figure 4 illustrates this dependency.

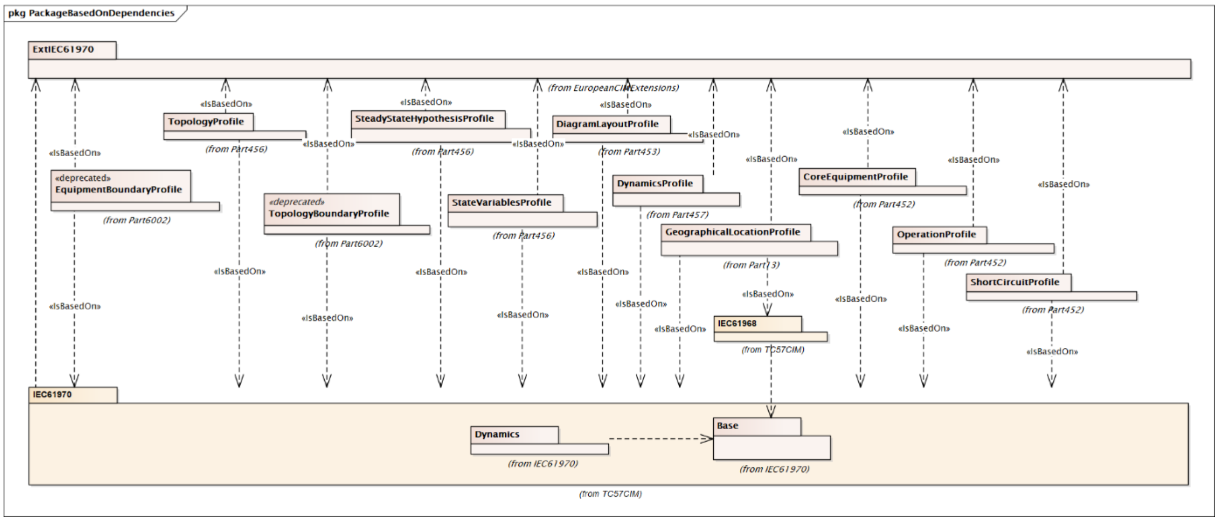


Figure 4 – Package dependency

4.4 Maintenance of UML

The maintenance of the UML is a key aspect for the proper handling of the extensions and the profiles. The maintenance of the canonical CIM is not covered in this section as it is part of the tasks of the IEC CIM Model Managers. This guide focuses on the procedures that are applied in ENTSO-E when maintaining the EA file. It is also important to note that in its present state the EA file contains the profile that are published under IEC 61970-400 series and a profile which is under IEC 61968, hence it is not only that ENTSO-E is involved in the maintenance of the EA file, but also IEC TC57 WG13. In addition, the CGMES is also on the way to be an international standard (IEC 61970-600-1 and IEC 61970-600-2). Therefore, the procedure described in this section will need to be aligned with IEC TC57 WG13.

The following main rules needs to be followed:

- The name of the EA file needs to be changed every time the file is posted. Strict version control shall be applied meaning also respective version classes in the EA file shall be updated.
- Before providing the EA file for further use the change log needs to be updated and information on the changes provided.
- A comparison needs to be run in EA in order to be sure that the changes reported in the resulting (updated) EA file are the desired one. This also helps to verify if the change log is correct.
- Before using any UML from IEC TC57 there shall be a good understanding what changes were applied compered to previous versions. In case of a doubt on the correctness CIM Model Managers need to be contacted.
- jCleanCim shall be run for model validation and the report shall be attached to the zip package in which the EA file, the change log is archived.

For details about the maintenance of the profiles please refer to section 7.2.3.

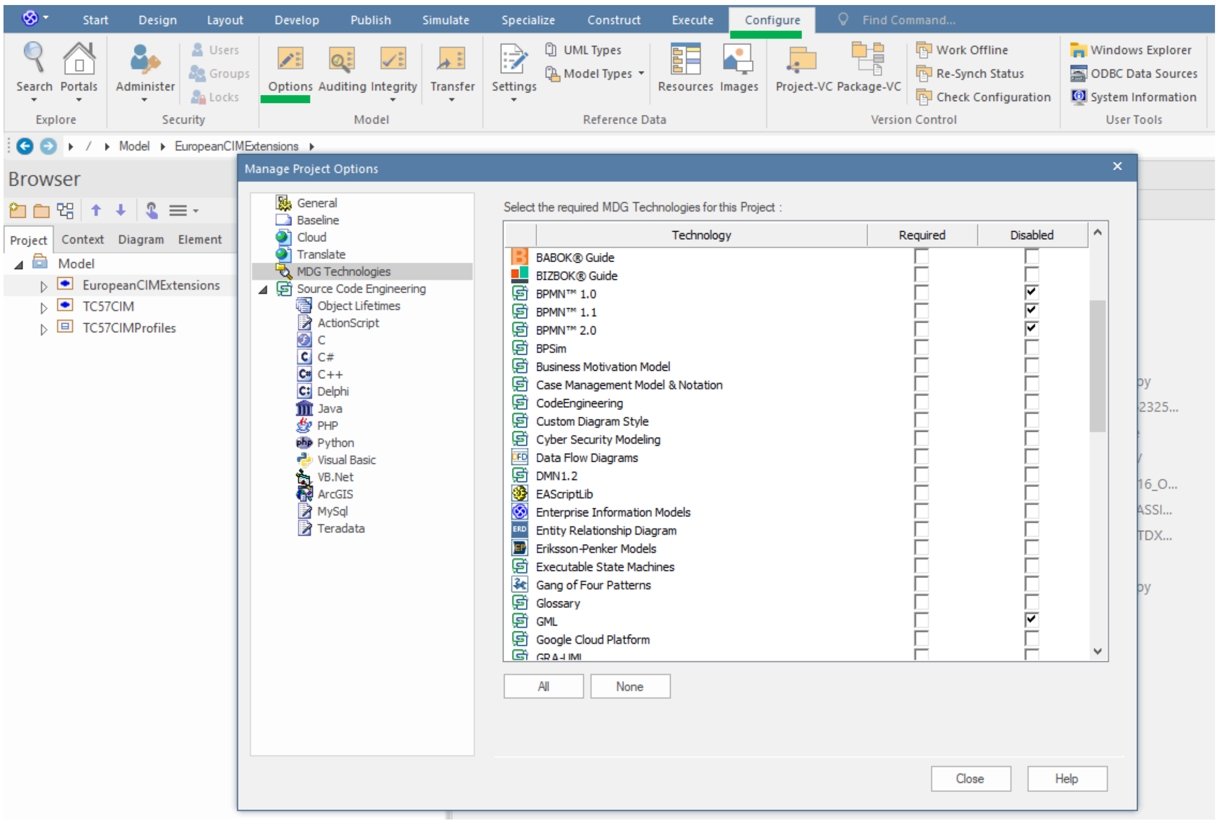
4.5 Parameters for Enterprise Architect

4.5.1 MDG Technologies

There are four technologies that needs to be disabled in EA. These are: BPMN 1.0, BPMN 1.1, BPMN 2.0 and GML. If these are enabled there are issues with the usage of the stereotypes.

Normally these settings are saved in the EA file and they do not need to be modified every time, however the settings need to be checked, especially when XMI is imported (i.e. in the process of moving the profiles/extensions to another EA file) or before the profiling work is started.

The “MDG Technologies” can be opened from Configure->Options as shown on Figure 5.



Select the required MDG Technologies for this Project :

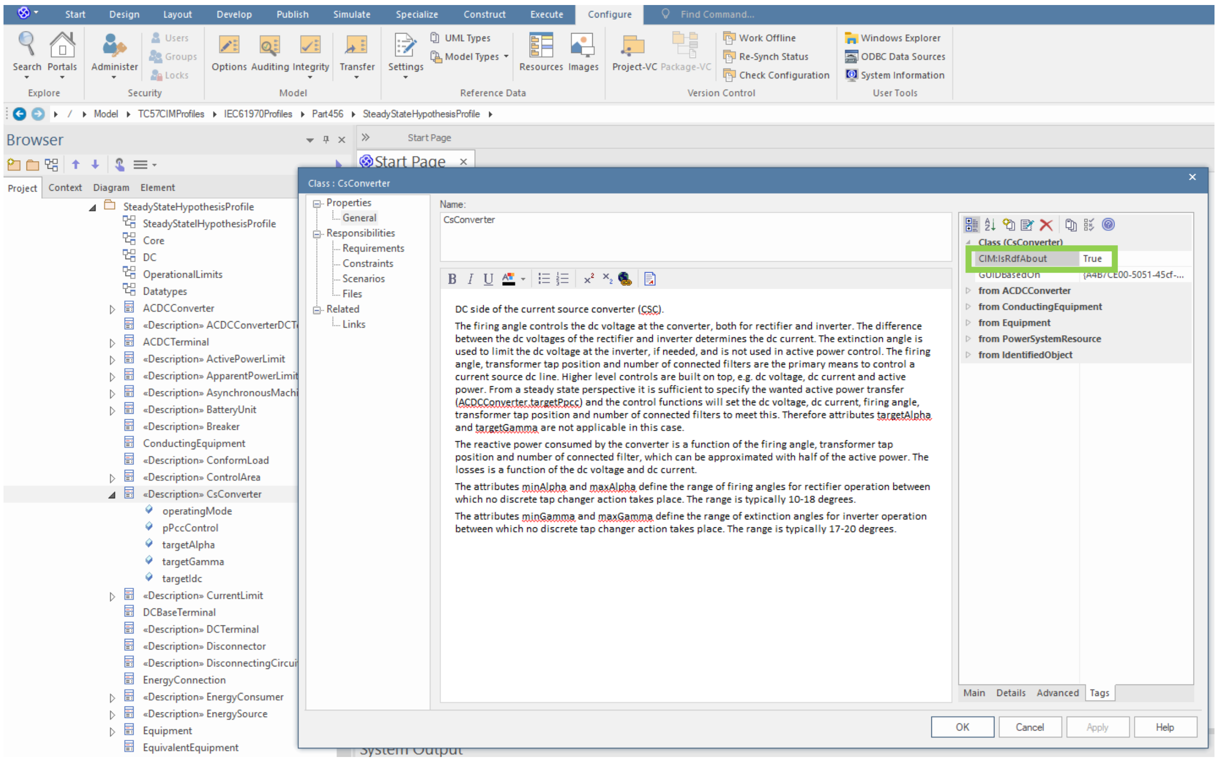
Technology	Required	Disabled
BABOK® Guide	☐	☐
BIZBOK® Guide	☐	☐
BPMN™ 1.0	☐	✓
BPMN™ 1.1	☐	✓
BPMN™ 2.0	☐	✓
BPSim	☐	☐
Business Motivation Model	☐	☐
Case Management Model & Notation	☐	☐
CodeEngineering	☐	☐
Custom Diagram Style	☐	☐
Cyber Security Modeling	☐	☐
Data Flow Diagrams	☐	☐
DMN1.2	☐	☐
EAScriptLib	☐	☐
Enterprise Information Models	☐	☐
Entity Relationship Diagram	☐	☐
Eriksson-Penker Models	☐	☐
Executable State Machines	☐	☐
Gang of Four Patterns	☐	☐
Glossary	☐	☐
GML	☐	✓
Google Cloud Platform	☐	☐
GRA-UML	☐	☐

Figure 5 – EA options

4.5.2 Reference data

When doing the profiling related to CGMES a convention to use the stereotypes for distinguishing extensions for main canonical model is applied. Also stereotype “Description” is currently used in Steady State Hypothesis (SSH) profile to indicate that the class is exchanged with rdf:about instead of rdf:ID. Besides the stereotype “IsBasedOn” is used by CimConteXtor and needs to be created as a reference data.

In addition the CGMES uses tagged values such as CIM:IsRdfAbout to express the same feature as expressed with the stereotype “Description”. This approach shall be considered temporal due to ongoing discussions in WG13. Figure 6 shows the tagged value in an attribute.



Name: CsConverter

DC side of the current source converter (CSC).

The firing angle controls the dc voltage at the converter, both for rectifier and inverter. The difference between the dc voltages of the rectifier and inverter determines the dc current. The extinction angle is used to limit the dc voltage at the inverter, if needed, and is not used in active power control. The firing angle, transformer tap position and number of connected filters are the primary means to control a current source dc line. Higher level controls are built on top, e.g. dc voltage, dc current and active power. From a steady state perspective it is sufficient to specify the wanted active power transfer (ACDCConverter.targetPpcc) and the control functions will set the dc voltage, dc current, firing angle, transformer tap position and number of connected filters to meet this. Therefore attributes targetAlpha and targetGamma are not applicable in this case.

The reactive power consumed by the converter is a function of the firing angle, transformer tap position and number of connected filter, which can be approximated with half of the active power. The losses is a function of the dc voltage and dc current.

The attributes minAlpha and maxAlpha define the range of firing angles for rectifier operation between which no discrete tap changer action takes place. The range is typically 10-18 degrees.

The attributes minGamma and maxGamma define the range of extinction angles for inverter operation between which no discrete tap changer action takes place. The range is typically 17-20 degrees.

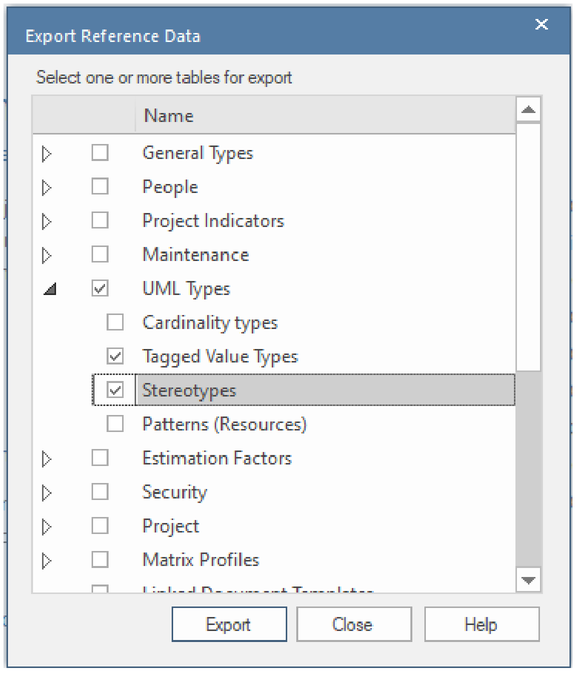
Class (CsConverter)
CIM:IsRdfAbout: True
GUIDBasedOn: {A4B7CE00-5051-45cf-...
from ACDCConverter
from ConductingEquipment
from Equipment
from PowerSystemResource
from IdentifiedObject

Figure 6 – Used tagged values

Therefore, it is important to have the reference data completed before the profiling work is started. There are two ways how to complete reference data:

1) Automatic process: Export reference data from another EA file and then import it

- Open the EA file that contains the right reference data.
- Select *Configure -> Transfer -> Export Reference Data*, which will show the dialog as in Figure 7. In this dialog select “Tagged Value Types” and “Stereotypes”. Then click Export and save the file so that it is ready for future import.



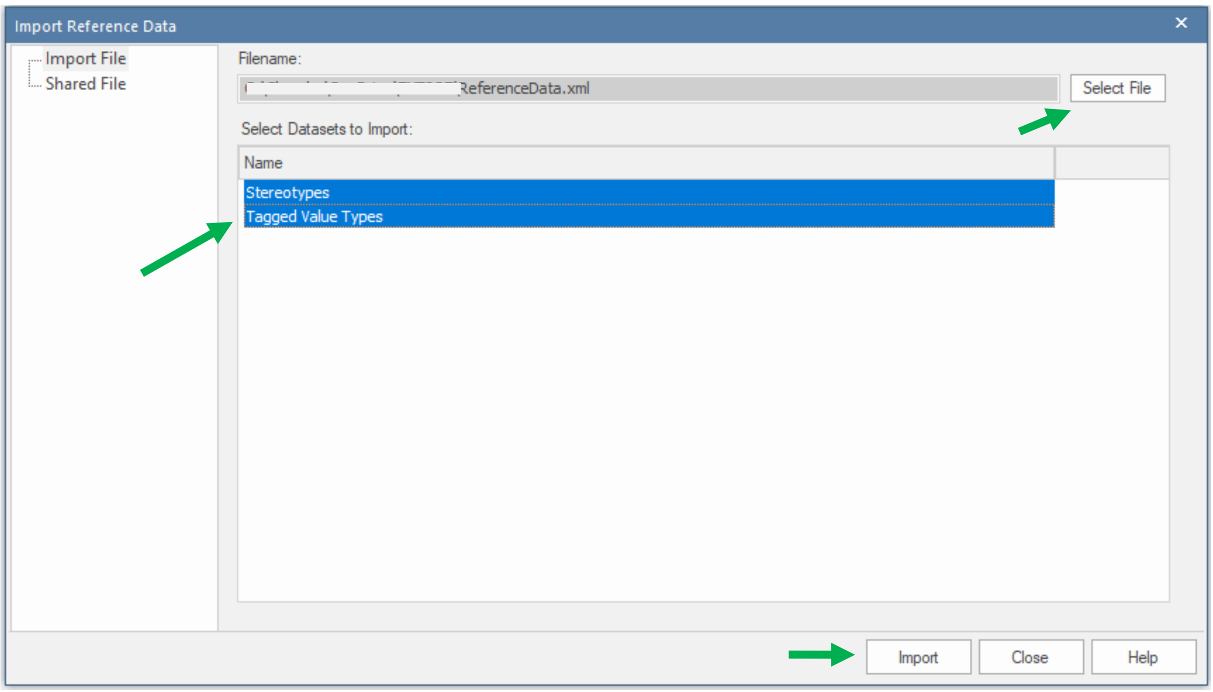
Select one or more tables for export

Name
☐ General Types
☐ People
☐ Project Indicators
☐ Maintenance
☑ UML Types
☐ Cardinality types
☑ Tagged Value Types
☑ Stereotypes
☐ Patterns (Resources)
☐ Estimation Factors
☐ Security
☐ Project
☐ Matrix Profiles

Figure 7 – Export Reference Data dialog

- Close the EA file from which the reference data was exported.

- Open the EA file where the reference data shall be imported.
- Select *Configure -> Transfer -> Import Reference Data*, which will show a dialog where three actions needs to be performed: 1) Click on button “Select File” to select the .xml file with reference data that was previously exported; 2) Select the two datasets to be imported, i.e. “Stereotypes” and “Tagged Values Types” as shown on Figure 8; 3) Click button “Import”.



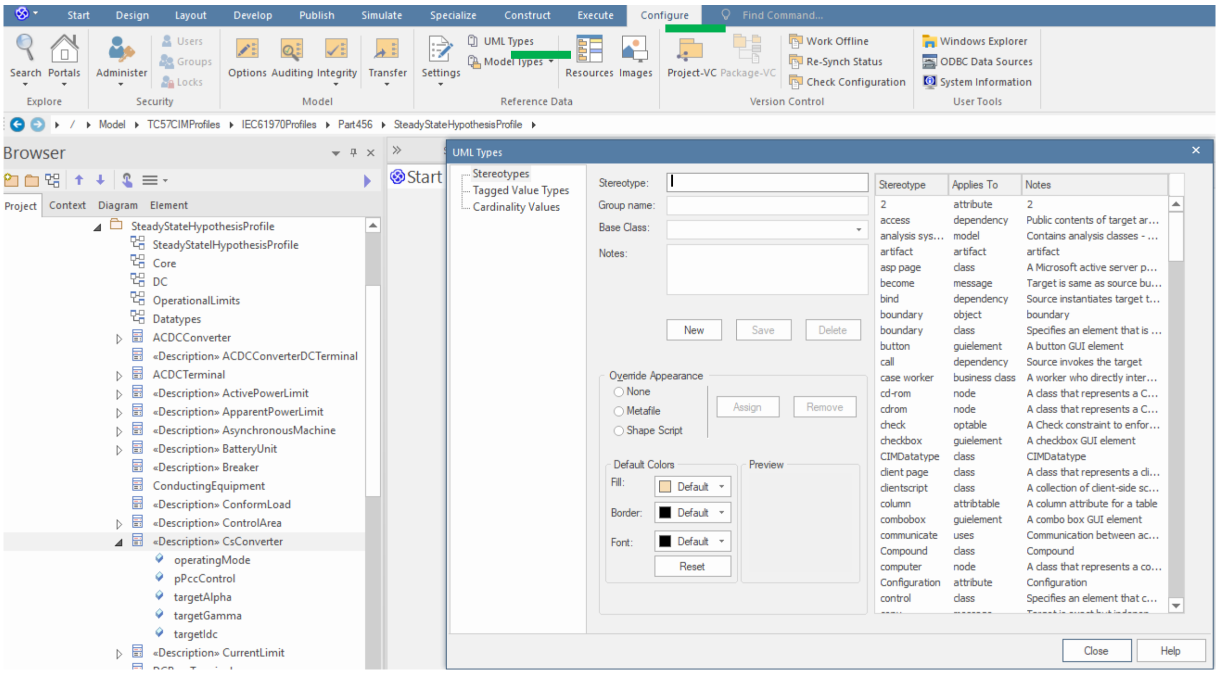
Filename: [redacted]ReferenceData.xml

Select Datasets to Import:

Name
Stereotypes
Tagged Value Types

Figure 8 – Import Reference Data dialog

- Imported stereotypes can be checked in the dialog “UML types” that can be opened using *Configure -> UML Types* as shown in Figure 9.



Stereotype	Applies To	Notes
2	attribute	2
access	dependency	Public contents of target ar…
analysis sys…	model	Contains analysis classes - …
artifact	artifact	artifact
asp page	class	A Microsoft active server p…
become	message	Target is same as source bu…
bind	dependency	Source instantiates target t…
boundary	object	boundary
boundary	class	Specifies an element that is …
button	guielement	A button GUI element
call	dependency	Source invokes the target
case worker	business class	A worker who directly inter…
cd-rom	node	A class that represents a C…
cdrom	node	A class that represents a C…
check	optable	A Check constraint to enfor…
checkbox	guielement	A checkbox GUI element
CIMDatatype	class	CIMDatatype
client page	class	A class that represents a cli…
clientscript	class	A collection of client-side sc…
column	attribtable	A column attribute for a table
combobox	guielement	A combo box GUI element
communicate	uses	Communication between ac…
Compound	class	Compound
computer	node	A class that represents a co…
Configuration	attribute	Configuration
control	class	Specifies an element that c…

Figure 9 – Imported reference data

2) Manual process:

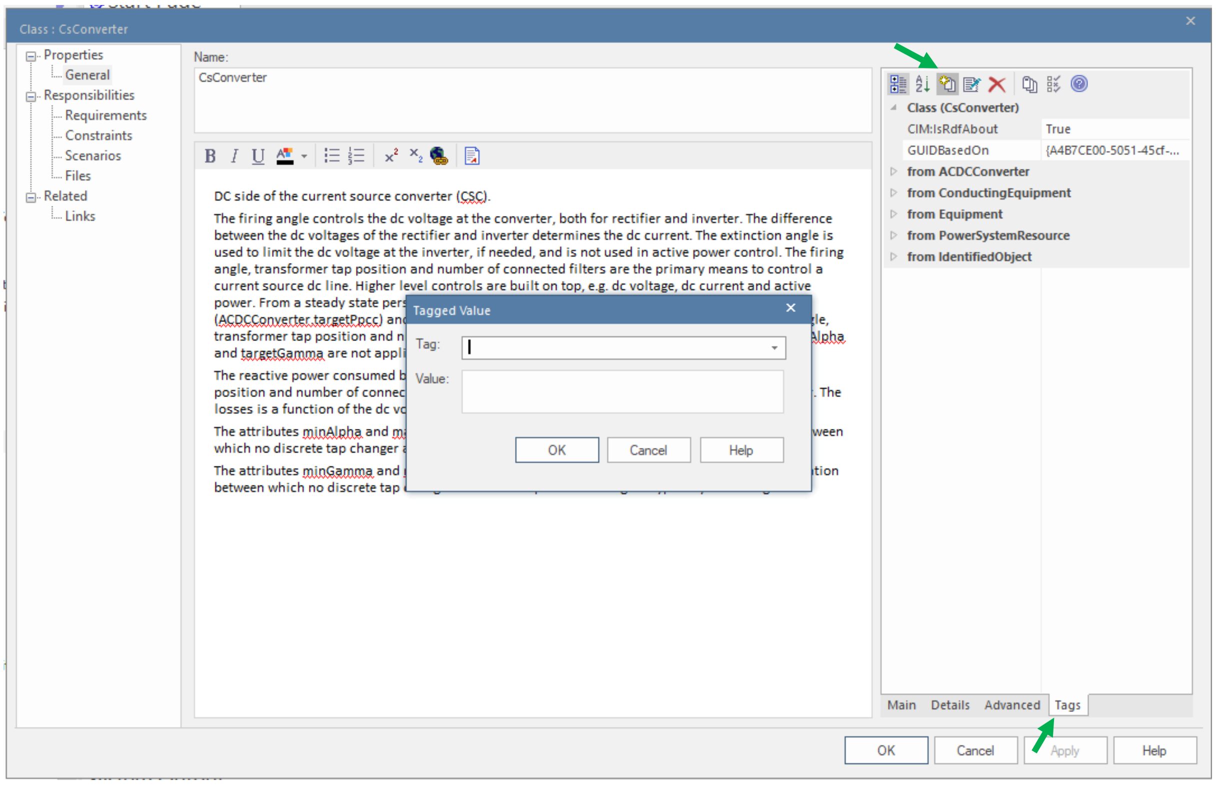
In manual process stereotypes are added by using the dialog shown in Figure 9. The field “Base Class” indicates for which entity the stereotype will be applicable. For instance, in case an extension related stereotype “European” needs to be added there shall be three stereotypes defined:

- one for a Class,
- one for an Attribute
- and one for an Association

so that this stereotype can be applied to any of these elements. The name of the stereotype is defined in the field “Stereotype”.

For tagged values if they are added manually this should be done directly in the attribute as shown on Figure 10. The Properties dialog of the class shall be opened. The tagged values are under the tab “Tags” where tagged values can be deleted or new tagged values can be added. For each tagged value a “Tag” and a “Value” shall be defined. For general setting of the tagged value CIM:IsRDFAbout the utility of the CimConteXtor shall be used (see section 0).

Note: It is highly recommended that this approach is used only in limited number of situations.



Name:
CsConverter

DC side of the current source converter (CSC).

The firing angle controls the dc voltage at the converter, both for rectifier and inverter. The difference between the dc voltages of the rectifier and inverter determines the dc current. The extinction angle is used to limit the dc voltage at the inverter, if needed, and is not used in active power control. The firing angle, transformer tap position and number of connected filters are the primary means to control a current source dc line. Higher level controls are built on top, e.g. dc voltage, dc current and active power. From a steady state pers (ACDCConverter.targetPpcc) an gle, transformer tap position and n Alpha and targetGamma are not appli

The reactive power consumed b position and number of connec . The losses is a function of the dc vo

The attributes minAlpha and m ween which no discrete tap changer

The attributes minGamma and tion between which no discrete tap

Tag:

Value:

Class (CsConverter)

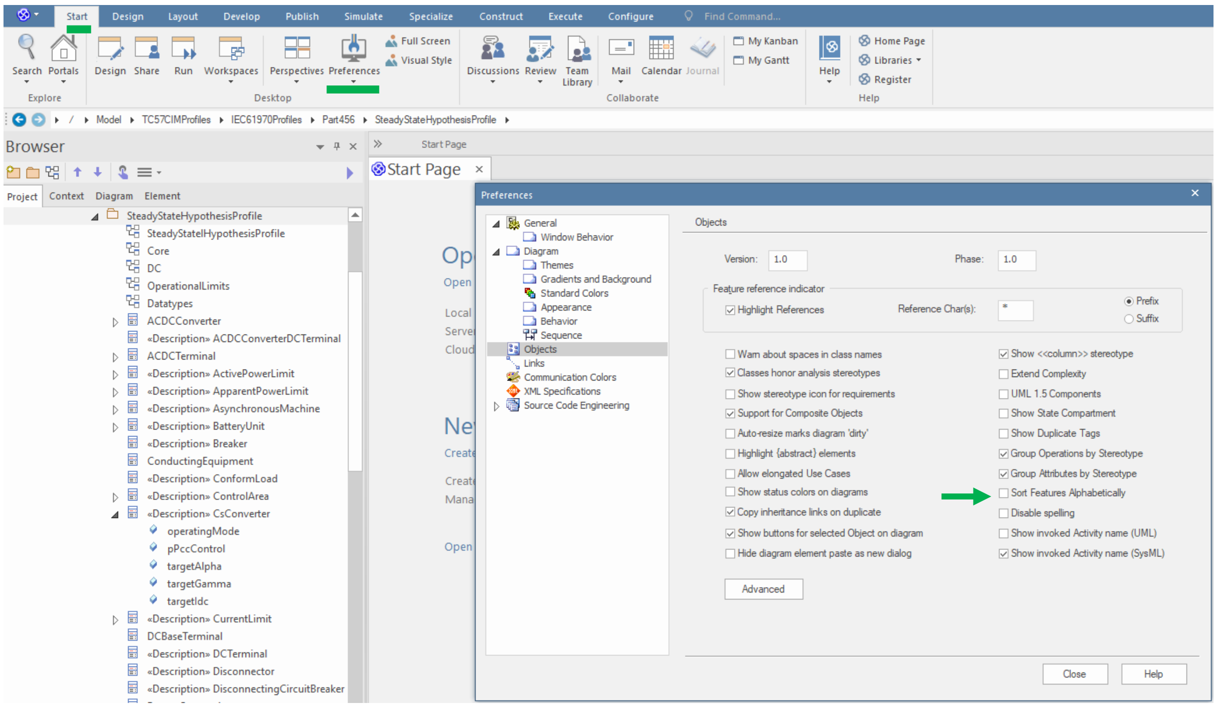
CIM:IsRdfAbout	True
GUIDBasedOn	{A4B7CE00-5051-45cf-...

from ACDCConverter
from ConductingEquipment
from Equipment
from PowerSystemResource
from IdentifiedObject

Figure 10 – Manual creation of a tagged value

4.5.3 Order of attributes in classes

In the CIM the classes and the attributes are ordered in a “business oriented” way and not necessarily in an alphabetical order. Thus, the “standard” setting of Enterprise Architect is to be modified by unchecking the option “Sort Features Alphabetically” as shown in Figure 11. The dialog “Preferences” opened from *Start -> Preferences*.



Objects

Version: 1.0 Phase: 1.0

Feature reference indicator

[✓] Highlight References Reference Char(s): * [● Prefix] ○ Suffix

[ ] Warn about spaces in class names
[✓] Classes honor analysis stereotypes
[ ] Show stereotype icon for requirements
[✓] Support for Composite Objects
[ ] Auto-resize marks diagram 'dirty'
[ ] Highlight {abstract} elements
[ ] Allow elongated Use Cases
[ ] Show status colors on diagrams
[✓] Copy inheritance links on duplicate
[✓] Show buttons for selected Object on diagram
[ ] Hide diagram element paste as new dialog
[✓] Show <<column>> stereotype
[ ] Extend Complexity
[ ] UML 1.5 Components
[ ] Show State Compartment
[ ] Show Duplicate Tags
[✓] Group Operations by Stereotype
[✓] Group Attributes by Stereotype
[ ] Sort Features Alphabetically
[ ] Disable spelling
[ ] Show invoked Activity name (UML)
[✓] Show invoked Activity name (SysML)

Advanced

Close Help

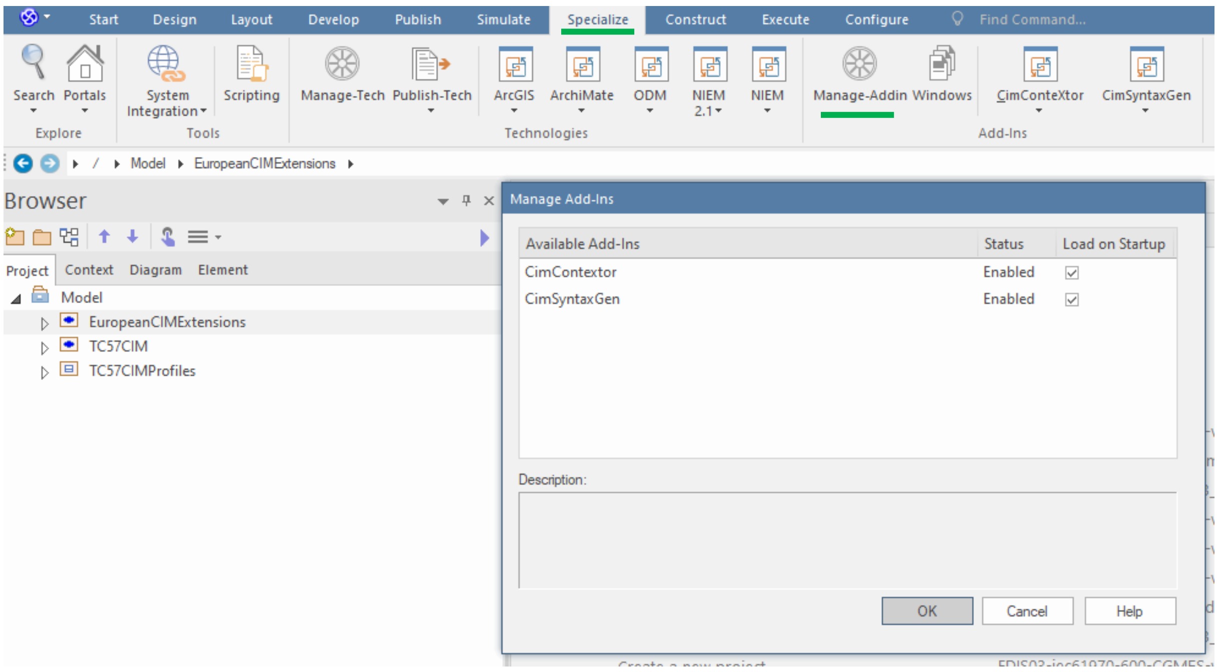
Figure 11 – Modification of the Sort Features Alphabetically

4.6 CimConteXtor and CimSyntaxGen plugins

Currently two Enterprise Architect add-ins are used in order to develop and maintain the profiles. CimConteXtor is used to perform the profiling work and CimSyntaxGen is used to generate the HTML documentation, RDF schema and SHACL constraints.

More information on these add-ins is available in [1] and [2]. During the installation procedure of the add-ins their respective configuration files are created in the respective folders under ….:\Users\{user}\AppData\Roaming\Zamiren. Examples of these configuration files are provided in section 0. It is important to make sure that the configuration files are correct for the task. This is especially valid for CimSyntaxGen. More details on this are provided in section 0.

In order to check if the installation was successful, click on the “Specialize” and then to “Manage-Addin” as shown in Figure 12. Both add-ins shall be shown as enabled in the “Manage Add-ins dialog”. Also, they shall appear in the toolbar.



Available Add-Ins	Status	Load on Startup
CimContextor	Enabled	✓
CimSyntaxGen	Enabled	✓

Figure 12 – CimConteXtor and CimSyntaxGen checking installation

4.7 jCleanCim

jCleanCim is an open source tool provided under terms of GNU LGPLv3 license (http://www.tanjakostic.org/jcleancim). It is developed to support validation and documentation generation from Enterprise Architect CIM and IEC 61850 UML models. It is a Java application, but (for some tasks) platform dependent due to usage of applications available on MS Windows only - Enterprise Architect and MS Word.

A tutorial for jCleanCim is available here: http://www.tanjakostic.org/jcleancim/jCleanCim-02v03/jCleanCimIntro.pptx

Rules to extend Canonical CIM

5.1 Overview

This section is including most of the rules defined in the CIM Model Management technical report, which shall be respected. However, note that this document is still not finalised within IEC. It is intended that these rules are also part of IEC 61970-401. When this standard is released this user guide shall be updated to refer to IEC 61970-401 and avoid duplication.

In order to properly extend canonical CIM it should be clear what a profile is. In a fairly simplistic view a profile can be defined as follows:

- It restricts what classes from the overall canonical model can be exchanged;
- It restricts what associations and attributes can be exchanged;
- It places additional restrictions on what data is required and other rules around this (e.g. limits, conditional requirements).

Note: Therefore, extensions have to take place in the canonical model first before they can be restricted at a profile level, i.e. it is not allowed to add extensions directly at the profile level.

There are different ways how to add extensions to the canonical CIM and they seem to fit in two categories:

The approaches to doing extensions seem to boil down to a choice:

- Extension approaches that make it easy to generate code, artefacts or schema;
- Extension approaches that make it easy to generate documentation.

Depending on the choice and the design of the software various parties can have different preferences on what to put focus on. For instance, some may like multiple-inheritance as it allows for automated generation of code, schema, APIs, etc. If there is a need to extend ConnectivityNode with an attribute myAttribute, doing this with multiple inheritance means that another class called ConnectivityNode is created, but in a completely separate package (on to which a tag denoting if it as an extension and the namespace URI for all extensions within this package) is added. On this new class (e.g. ext:ConnectivityNode) the attribute myAttribute is added and the original class (cim:ConnectivityNode) inherit from ext:ConnectivityNode as well as cim:IdentifiedObject. The benefit here is that an instantiated cim:ConnectivityNode class now includes all its CIM attributes and the attributes from ext:ConnectivityNode. If data from another system is received that has no knowledge of the extensions, they can provide a ConnectivityNode and it will be created as exactly the same class as in the receiving system ConnectivityNodes (where the extensions was done).

If the approach of creating MyConnectivityNode (as a separate class to extend ConnectivityNode) is taken the effect is either:

- Forcing the creator of the data to create ext:MyConnectivityNode instances rather than cim:ConnectivityNode – thus breaking the data for any systems that have no knowledge of the extensions;
- Forcing development of an internal logic that has to merge the two classes together during the code/artefact generation process, so nobody ever sees an instance of MyConnectivityNode.

The big issue with the second approach (i.e. forcing development of an internal logic that has to merge the two classes together) is that it’s a completely new, custom way of doing things. With multiple inheritance existing off-the-shelf tools can be used. Code (whether it be Java, C++ etc.) can all be generated and work out-of-the-box. Any tooling that works with meta-models/UML can deal with the extensions automatically (e.g. any tool that builds a user interface from the model would show myAttribute as being a valid attribute for cim:ConnectivityNode as it can determine its attributes from its parent classes. Using a special custom approach of special associations or subclassing would require the receiver to know about how the sender is doing this.

Managing these extensions is key aspect. It needs to ensure that any new parent class goes under a different package, so it can be identified very quickly, which classes are standard and which are extended. It also has the benefit of allowing multiple separate extensions, e.g. it could be two packages with ext:ConnectivityNode and my:ConnectivityNode, then cim:ConnectivityNode inherits from both.

This also means that any attributes or associations will be correctly marked as being an extension from the appropriate package by looking at which class they are in (or connect to). Managing upgrades to the model involves looking through the extension packages and identifying where something has changed or is now broken (either manually or automatically).

Taking into account the background information above and the present tooling provided by Enterprise Architect and related add-ins the following options of extensions can be considered. Note that this include options that are not allowed at the moment. They are included to have an example and explanation what is not allowed.

1) **Option 1:** Multiple inheritance in the extension package, “implicit” inheritance in the profile.

This is the current approach applied for extending attributes on existing classes.

- Extension package

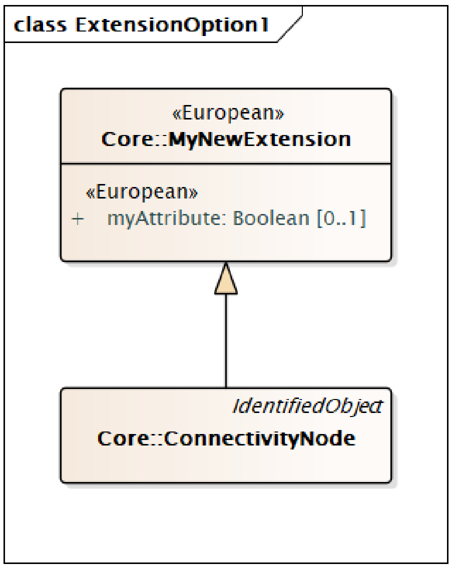


Figure 13 – Extension option 1 – extension package

- Profile

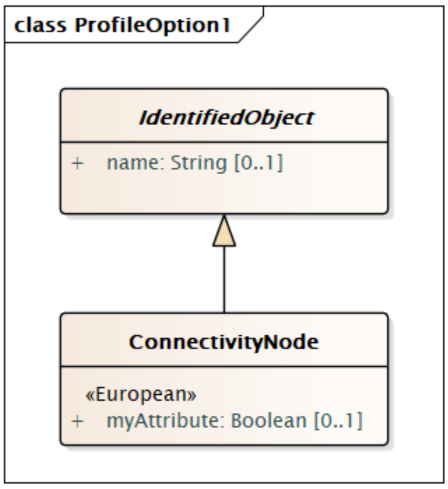


Figure 14 – Extension option 1 – profile package

- Instance data

```
<cim:ConnectivityNode rdf:ID="_8a5a6e86-55e4-476b-a3bd-7832e6168569">
    <cim:IdentifiedObject.name>XMO_TE32</cim:IdentifiedObject.name>
 <eu:ConnectivityNode.myAttribute>true</eu:ConnectivityNode.myAttribute>
  </cim:ConnectivityNode>
```

2) **Option 2:** Multiple inheritance in the extension package, “explicit” multiple inheritance in the profile (**not allowed option**, *for illustration only*)

- Extension package (same as option 1)

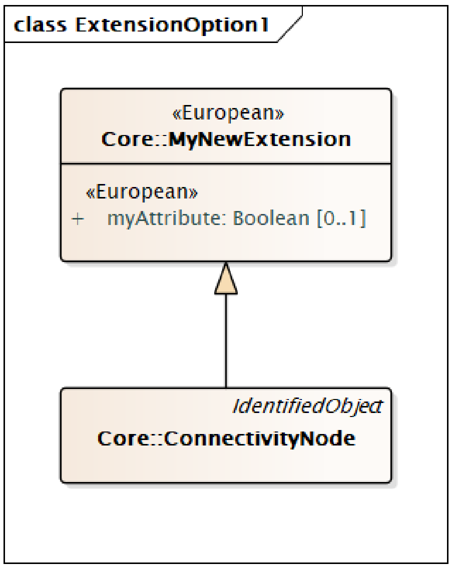


Figure 15 – Extension option 2 – extension package

- Profile

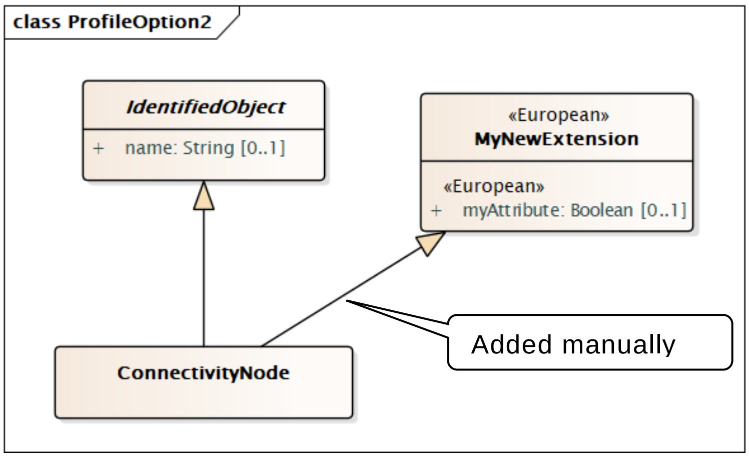


Figure 16 – Extension option 2 – profile package

Note that CimConteXtor does not offer that ConnectivityNode inherits also from MyNewExtension. The inheritance could be added manually (outside CimConteXtor) in the profile, however **this is not allowed**.

- Instance data

```
<cim:ConnectivityNode rdf:ID="_8a5a6e86-55e4-476b-a3bd-7832e6168569">
  <cim:IdentifiedObject.name>XMO_TE32</cim:IdentifiedObject.name>
<eu:MyNewExtension.myAttribute>true</eu:MyNewExtension.myAttribute >
 </cim:ConnectivityNode>
```

3) **Option 3:** Subtyping/subclassing (allowed option)

- Extension package

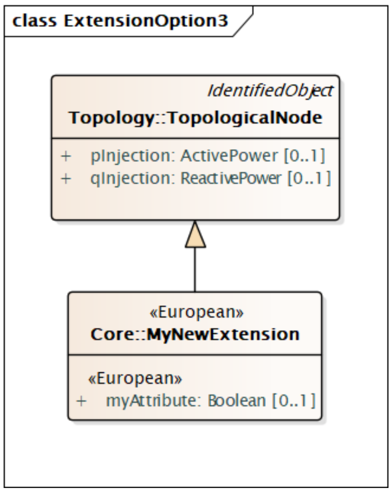

class ExtensionOption3
IdentifiedObject
Topology::TopologicalNode
+ pInjection: ActivePower [0..1]
+ qInjection: ReactivePower [0..1]
«European»
Core::MyNewExtension
«European»
+ myAttribute: Boolean [0..1]

Figure 17 – Extension option 3 – extension package

- Profile

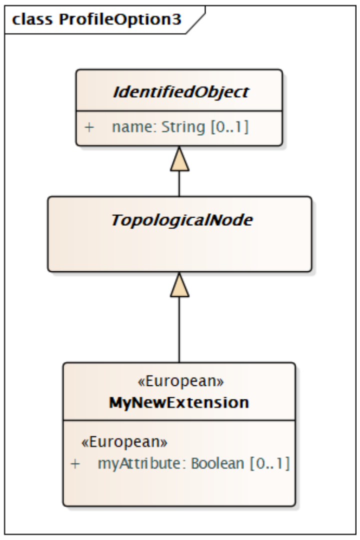


Figure 18 – Extension option 3 – profile package

- Instance data

<eu:MyNewExtension rdf:ID="_8a5a6e86-55e4-476b-a3bd-7832e6168569">
<cim:IdentifiedObject.name>XMO_TE32</cim:IdentifiedObject.name>
<eu:MyNewExtension.myAttribute>true</eu:MyNewExtension.myAttribute >
</ eu:MyNewExtension>

4) **Option 4**: Add the extension higher in the inheritance hierarchy (to avoid multiple inheritance).

The goal is to extend Equipment with one attribute. **This option is not really practical** as pollutes the profile and creates other issues. Therefore, it is not allowed option.

- Extension package

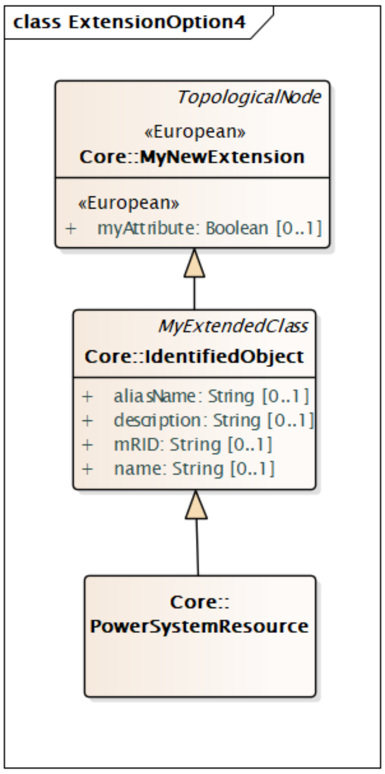


Figure 19 – Extension option 4 – extension package

- Profile

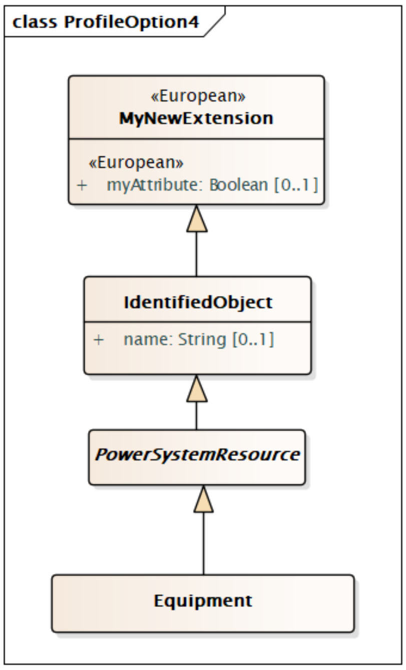


Figure 20 – Extension option 4 – profile package

- Instance data

<cim:Equipment rdf:ID="_8a5a6e86-55e4-476b-a3bd-7832e6168569">
<cim:IdentifiedObject.name>XMO_TE32</cim:IdentifiedObject.name>
< eu:MyNewExtension.myAttribute>true</eu:MyNewExtension.myAttribute >
</cim: Equipment >

5) **Option 5:** Use association (allowed option).

- Extension package

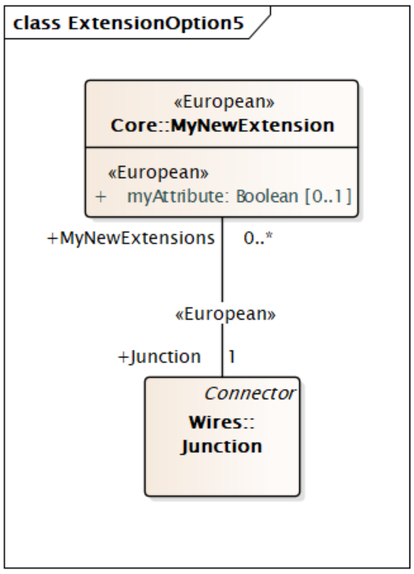


Figure 21 – Extension option 5 – extension package

- Profile

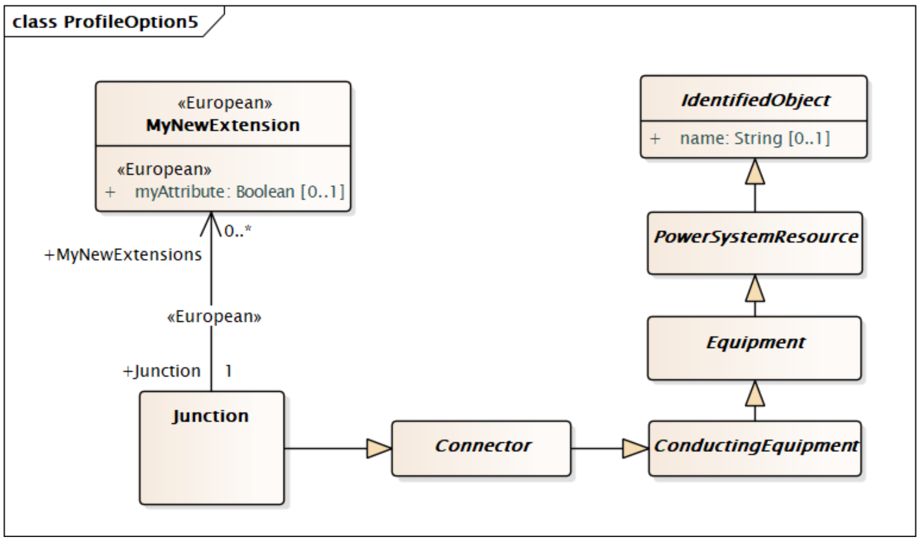


Figure 22 – Extension option 5 – profile package

- Instance data

```
<cim:Junction rdf:ID="_8a5a6e86-55e4-476b-a3bd-7832e6168569">
  <cim:IdentifiedObject.name>XMO_TE32</cim:IdentifiedObject.name>
<eu:Junction.MyNewExtensions rdf:resource="#_0b5bf2ef-a008-4b9a-9fe9-3c528c8bec50" />
</cim:Junction>

 <eu:MyNewExtension rdf:ID="_0b5bf2ef-a008-4b9a-9fe9-3c528c8bec50">
  <eu:MyNewExtension.myAttribute >True</eu:MyNewExtension.myAttribute>
 </ eu:MyNewExtension >
```

5.2 General rules

The following rules shall be respected:

a) All extensions shall be packaged in a separate package in EA file outside the TC57 CIM and outside the package with the profiles (see Figure 3). When extending CIM with new packages:
   - Use packages to create manageable and logical model pieces.
   - Before creating a new package, do a search for the intended package name; if such a package name already exists in the canonical model, consider using another name or adding a prefix to the name of the new package.
   - Non-standard extensions should use a new namespace to indicate the source of the extensions and that they are not standard CIM.

b) In cases where the objective is to extend a class with new attributes, the approach using multiple inheritance in the extension package (see option 1 in section 5.1) shall be used. The added class and the attributes shall be stereotyped.

c) In cases where the objective is to extend a class with new association, the approach used in option 5 in section 5.1 shall be used. The association shall be stereotyped.

d) In cases where the objective is to extend an enumeration, a new enumeration is created. The new enumeration includes the added enumerated values and the existing attributes from canonical CIM. The attributes of the enumeration are not stereotyped. The new enumeration class is stereotyped.

e) When extending the CIM with new classes or Associations:
   - Avoid creating new primitive data types. Reason is that CIM is a semantic model that stays away from the extensive data typing available in many implementation technologies.
   - Avoid creating objects that do not represent objects in the CIM domain.
   - Before creating a new class, do a search for intended class name; if such a class already exists in the model, consider using another name or adding a prefix to the name of the new class.
   - Create new domain object classes by sub classing (also known as subtyping) an existing class, e.g. IdentifiedObject, PowerSystemResource, Asset, Document, ActivityRecord, etc. If it is not possible to subclass the object, an association between the two objects may be created. When adding an association, the information from the use case that motivated the addition is typically lost. A way to avoid this loss is to include the information in the documentation of the attribute or association end. Stereotypes or tagged values are another possibility but no commonly agreed way to do this exists yet and therefore should be avoided.
   - When defining a new subclass, display on a diagram the superclass and its related classes with all their attributes to avoid inadvertent duplication: (a) either with the same attribute/association end names that get inherited, or (b) by duplicating an existing concept but with different name.
   - If the existing description is ambiguous or not clear, immediately file a CIM issue. For ENTSO-E CIM issue is submitted to cim@entsoe.eu.
   - Linkages between packages should be carefully considered.

5.3 Class rules

When creating a new class, the following shall be respected (according to CIM Model Manager rules):

- The following UML class features are used in CIM:
   - Name
   - Documentation
   - Stereotype
- In CIM, a class is used to describe either domain entities or various data types specific to the CIM domain:
   - <<Primitive>> Primitive data types as Booleans, Float, String etc. Primitive cannot have attributes.
   - <<CIMDatatype>> Simple data types specific for the CIM, e.g. ActivePower, Resistance etc. CIMDatatype has at least three attributes: value, unit and multiplier. The "value"

attribute is of Primitive type, and unit and multiplier are of an enumeration type. When required additional attributes can be defined to describe for example “demoninatorUnit” and “denominatorMultiplier”, but only one “value” attribute is allowed.

- <<Compound>> Compound data types specific for the CIM, e.g.StreetAddress etc. Compound has no identity and is a simple group of related attributes, Compound may have attributes whose types are Primitive, enumeration, CIMDatatype or Compound. A circular dependency among Compound types is to be avoided.
- <<enumeration>> Enumerations, e.g. UnitSymbol, UnitMultiplier, Currency, etc.

None of the above stereotypes. A domain object that participates in inheritance and/or association relationships with other domain objects. Most of domain objects inherit from the class IdentifiedObject and thus get the identifier and one or multiple names.

- CIM classes with a stereotype other than <<deprecated>> are types that never participate in relationships (i.e., no associations, no inheritance), but are used as types for attributes.

  Clarification: *The stereotypes given to extensions are an exception of this rule (as long as the stereotypes are used to identify the extensions).*
- CIM does not use abstract classes, because CIM is in itself an abstract model. In contrast, profiles may distinguish between concrete and abstract classes.
- CIM does not use association classes, for the sake of keeping the used subset of UML to the minimum that is well supported by tools and well understood by the wide community.
- CIM class names are unique among all TC57CIM class names without consideration of the package hierarchy.
- Classes should be ordered alphabetically or in order of importance or by logical grouping. We do not enforce alphabetically ordering automatically in the UML tool. The order of classes in the UML tool with be the order of classes printed in the IEC document.
- Documentation shall be complete for each new class.

5.4 Attribute rules

When creating an attribute, the following shall be respected (according to CIM Model Manager rules):

- The following UML attribute features are used in CIM:
  - Name, always used and must be unique among all levels of specialisation.
  - Type that is one of the stereotyped classes (Primitive, CIMDatatype, Compound, enumeration). When choosing type, ensure you select it from the list of existing types rather than simply type the text into Enterprise Architect. Attention on Primitives: use CIM Primitive datatypes (e.g., String, Boolean), not default UML ones (string, boolean).
  - Scope is always public.
  - Multiplicity is always [0..1], i.e. optional.
  - Initial value, if used, always denotes a constant for all instances of the class where the attribute belongs, and the attribute has to be set as both static and constant. This is only allowed for unitMultiplier and unitSymbol of CIMDatatype and “package version” attributes which hardcode the date and version attributes.
  - Documentation. The documentation should not use or rely upon special formatting. Plain text is assumed. It is suggested to avoid special characters here also.
- The attribute stereotype can be used to identify the attribute as an extension. Once the attribute is fully incorporated in the TC57 canonical model the stereotype is removed.
- Attribute names are unique within a classifier and Enterprise Architect will enforce this.
- The attribute order should normally be alphabetical unless there is some clear reason to group like attributes together or reorder. We do not enforce alphabetic ordering in the UML tool.

5.5 Enumeration rules

When creating an enumeration literal, the following shall be respected (according to CIM Model Manager rules):

- Enumeration literals are attributes within an <<enumeration>> type. The following UML attribute features are used in CIM:
    - Name, always used and must be unique among all levels of specialisation.
    - UML stereotype <<enum>>.
    - Enumeration literal has no type by definition.
    - Enumeration literal scope is public by definition.
    - Enumeration literal has no multiplicity by definition.
    - Initial value, if used, denotes a code that has semantic binding. If used, the code must be unique among all the codes for literals within the containing enumeration type. Also, either all or no literals within an enumeration type have a code.
    - Documentation. The documentation should not use or rely upon special formatting. Plain text is assumed. It is suggested to avoid special characters here also.
- Other than <<enum>> or <<deprecated>> attribute stereotype can be used to identify the enumeration as an extension. Once the attribute is fully incorporated in the TC57 canonical model the stereotype is removed.
- Like for attributes, literal names are unique within a classifier and Enterprise Architect will enforce this.
- There is no ordering required or reinforced; the order found in the UML model will be the order of printing.

5.6 Association rules

When creating an association, the following shall be respected (according to CIM Model Manager rules):

- Associations describe how classes are related. Only classes describing domain objects, i.e. classes without data type stereotypes such as <<enumeration>>, <<Primitive>>, <<Compound>>, or <<CIMDatatype>>, may participate in associations.
- An association has two ends referred to as association ends. CIM associations have unspecified direction but have both ends specified which means they are implicitly bi-directional (i.e., it is possible, through end names, to navigate from A to B and from B to A).
- The association description and association name are not specified. These become a maintenance problem if specified and also show up on diagrams by default. Instead, we specify the association end names and descriptions.
- The UML association features used in CIM are as follows:
    - Stereotype <<informative>>, only in case the association is informative. Other stereotypes are allowed to identify the extensions.
    - The UML association end features used in CIM are:
        - Association end name
        - Association end multiplicity
        - Association end description.

      No other options should be used, although some existing association ends in CIM are defined as aggregation (= shared).
    - Note that, depending on local user settings for EA tool:
        - it may happen to create by default a directed association, so ensure that the direction for association and both its ends is “Unspecified”;
        - if using aggregation from the toolbox, the diamond may be on unexpected side of relationship, so always create first a simple association.
- *To facilitate the correct dependency processing, draw an association from a class in the depending package (source end) towards a class in the dependent package (target end)*. NOTE: For example, an association between a class in IEC61968 and a class within IEC61970

would be drawn from (the source end of) the class defined within IEC61968 towards (the target end of) the class within IEC61970. This since IEC61968 depends upon the IEC61970 package.

- There appears to be no clear manner to control the association end ordering (within the context of a class) in the Enterprise Architect tool so this is an unmanaged aspect of the UML model. In fact, UML does not specify association ends are owned by the class, they are by default owned by the association itself.
- Association end names should be unique among all specialisations of a class. For example, if IdentifiedObject associates to the Name class with an end name of “Name”, no specialisation of IdentifiedObject may create another association whose end name is “Name” regardless of the class to which it associates.

  Note that any number of associations from other classes may use the same name (e.g. “IdentifiedObject”) as the association end name on the IdentifiedObject end of the association, as long as the prior rule of uniqueness among specializations is respected.

5.7 Diagram rules

When creating a diagram, the following shall be respected (according to CIM Model Manager rules):

- The following UML diagram features are used in CIM:
  - Name
  - Documentation
- Diagrams should not use shadows, colours or diagram frames when printing model documents.
- Diagrams should use A4 page size.
- Each package should typically include at least one diagram that contains all the classes in a package, and an arbitrary number of other diagrams.
- In general, when editing diagrams, it is possible to set show/hide attributes or relationships for a diagram, to display the desired subset of the model.
- Diagram names are used for model document generation and should be unique across the whole CIM model. The CIM UML tool documentation generation allows for the combination of containing package (unique in model) and diagram name to uniquely identify a diagram, but we strive to keep diagram names unique across all packages.
- Diagrams should be placed inside of packages (the normal case) or classes (e.g. in Dynamics package). The order of diagrams in the package is retained and should be from most general to specifics. Otherwise, diagrams should be in alphabetical order.

5.8 Multiplicity rules

When defining multiplicities, the following shall be respected (according to CIM Model Manager rules):

- Multiplicities are used to describe the number of expected objects at each end of an association as well as if an attribute is optional or mandatory.
- The following multiplicities are common (but others are allowed) for association ends in CIM:
  - 0..1 an object may or may not exist,
  - 1 an object always exists,
  - 0..* any number of objects may exist,
  - 1..* at least one object exists.
- Multiplicities shall be chosen to specify what can be expected in the domain. Here are some guiding examples:
  - An auto transformer will have at least one winding
  - A transformer with separate windings will have at least two windings
- Multiplicity minimum does not mean all profiles must include the association, but if a profile does include the association it should respect the minimum.

  Note that minimum multiplicities are normally not enforced by implementations at all times. For example, in the middle of a transaction one may require violation of minimum multiplicities.

5.9 Inheritance rules

When defining inheritance, the following shall be respected (according to CIM Model Manager rules):

- Inheritance is used to specialise an existing class. The inheriting class is more specific than the base class. Inheritance is the strongest possible dependency; it is often misused and should be used with care (note: everything that can be expressed through inheritance could also be expressed through composition).
- The CIM is currently using a single type, inheritance hierarchy and the use of multiple inheritance is not allowed.
- Inheritance relationship must be drawn from the more specific class to the more general class. In case the classes from two top-level packages are involved, the subclass must be in the dependent package, and inherit from the class in a package upon which it depends.
- Inheritance should never create situations where attribute names or role names are duplicated or "override" within the inheritance linage.

5.10 Documentation rules

When completing description of UML elements, the following shall be respected (according to CIM Model Manager rules):

- Make sure to add a description to the UML elements (package, diagram, class, attribute, association ends etc.) when creating them. The description shall explain the meaning of the entity as clearly as possible. It is often useful to search dictionaries and the web for good descriptions.
- Use full sentences with proper upper case start and ending period. These get put into IEC documents and must conform to standard editing of such a document. Any references to standards should be specified in a manner compatible with inclusion in IEC documents. The same is true of units, variable names, or other IEC editorial policies.
- Make sure to update the description every time you do a modification to existing element (e.g., when changing name of an association end or a class, do extended search in the overall model and ensure to update the in-text references to the element whose name changes).
- Document the element itself, not its related or contained elements. Details of attributes should be documented on the attributes themselves, rather than on the class that contains them.
- Document the purpose of the attribute. Avoid "This attribute is used ..." or similar.
- Document the purpose of association end. Avoid "A is related to one or more B" since UML shows it already.
- Avoid using CIM class names in the description/notes. For example, prefer "usage point" over "UsagePoint". In some rare cases an explicit reference to an attribute may be required for clarity (e.g., "phase information is available in 'Terminal.phases'."), but such references create a maintenance problem.
- Avoid using mark-up in the documentation (e.g., bold, lists, superscripts). Note this means avoid using variable names in equations in the text to also conform to the IEC documentation rules which require italic type for variable names.

5.11 "deprecated" stereotype rules

- This stereotype is recognised by the CIM UML validation and document generation tool and can be applied to any of the UML concepts defined below. The typical usage of the <<deprecated>> stereotype is for the purpose of preserving backwards compatibility for the normative, already published content, during a release or two, while indicating to the users that the item is likely to actually be removed in the future. This is a graceful means of phasing out obsolete or re-factored elements, and leaving some time to the users to provide implementation in terms of the new features replacing those marked with <<deprecated>>.
- It is possible and supported by the CIM UML tool to have multiple stereotypes on an element (e.g. <<deprecated, CIMDatatype>>).
- The <<deprecated>> stereotype may be used on attributes, associations, classes, packages, and diagrams in the UML model. It is recommended when deprecating a package to deprecate all its contained classes as well. When deprecating a class, it is recommended to deprecate all

associations in which it is involved and all its native attributes. Inheritance from a deprecated class is less clear and no general guidance is given, except to avoid this situation if possible.

5.12 "Inf" prefix rules

- "Inf" is the prefix used to denote informative CIM UML sub-packages. This is to avoid name clashes among normative and informative packages (e.g., 'Core' vs. 'InfCore'), and to make informative sub-packages obvious through their name. Note that this will apply to all packages that start with this prefix (e.g., 'InfCore' as well as 'Informative').
- The content of the informative package, including any sub-packages and its content, recursively, is considered as informative in the sense of IEC documents and is therefore by default excluded from the document generation. However, the CIM UML tool has an option to enable printing the informative content as well, which is useful for e.g., sharing documentation of the current work in an informative sandbox with the working group members, without impact to the stable, normative parts of the model.
- An informative package can be defined at any depth in the CIM UML model. The only requirement is the "Inf" prefix in the package name.
- It is recommended to specify "Inf" packages as "private" packages in Enterprise Architect package properties, so they can be filtered out of diagrams showing the sub packages.
- Associations between existing classes may be added in an informative package by stereotype of <<informative>> on the association.

  Note: Associations between two Informative classes need not be stereotyped, but care must be taken when moving classes to standard packages.

5.13 "Doc" prefix rules

- "Doc" is the prefix used to denote those CIM UML sub-packages that contain diagrams used for the IEC document template only, and that should not be printed with the content of the regular model. The sub-package of "ABC" with the name "DocABC" will be matched, and considered as informative package from the perspective of automatic document generation (i.e., it will be skipped), but it still can be referenced from the IEC document template to include the diagram, for instance in its introductory examples section.
- It is recommended to specify "Doc" packages as "private" packages in Enterprise Architect package properties, so they can be filtered out of diagrams showing the sub packages.

5.14 DetailedDiagrams rules

- It may be useful, for instance, to "park" temporary diagrams describing CIM issues during the resolution process, without ever interfering with the official IEC document generation. If there are diagrams that should not be printed as a part of the package documentation but that should be kept within a normative package because they belong there (e.g., describe an issue related to one or more normative classes), create a sub-package called "DetailedDiagrams" and put those temporary diagrams below it. The package name "DetailedDiagrams" is a reserved word, relevant for document generation, which allows the CIM UML tool to simply skip that package and its contained diagrams when generating documentation. This is the means to keep the informative or temporary diagrams within the normative packages, but to skip them during document generation.
- DetailedDiagrams package can be defined at any depth in the CIM UML model. The only requirement is the package name "DetailedDiagrams". Because this name may not be unique, it is not possible to reference diagrams under this package from within the IEC document template (and one will never need to do that, by definition).
- It is recommended to specify "DetailedDiagram" packages as "private" packages in Enterprise Architect package properties, so they can be filtered out of diagrams showing the sub packages.

5.15 Naming rules

5.15.1 General

- General rules for IEC TC57, available in "TC57 CIM Naming_Rules_Draft_R00_2008-05-16v3.doc", give the overall guidelines for naming conventions of UML models at any abstraction level. The rules defined in this technical report, dedicated to CIM UML information model, take precedence over the rules in the above document in case the rules differ.

- Use English language for names.
- Names shall describe the problem domain so it aids human understanding.
- Words in concatenated names use camel notation (UpperCamelCase or lowerCamelCase rule).
- Names contain alphanumeric characters only, i.e. [0..9, A..Z]. Space or special characters are not allowed, e.g.< > & _ - " ' ( ) [ ] { } ? ! ; : . , * etc. Historical exceptions are allowed for the underscore character.
- Names always start with a letter, a numeric value is not allowed.
- Make sure a new name for a package or class is unique within its package scope and preferably in the whole model to avoid any ambiguity.
- Make sure to reuse an existing attribute name for a new attribute (e.g. sequenceNumber instead of sequence or seqNum or seqNumber).

  Note: It is discouraged to distinguish names by upper and lower case changes alone. We have a few examples of this, but the practice is discouraged.

5.15.2 Package

- Package names start with upper case (UpperCamelCase rule). Package names must be unique across the whole CIM.

5.15.3 Class

- Class names start with upper case (UpperCamelCase rule). Class names should use singular form. Class names must be unique across the whole CIM. Be aware there is a potential conflict of profile names and class names that may occur from the 62321-100 standard for XSD naming rules, so the choice of profile names and class names should be made with this consideration.

5.15.4 Enumeration class

- Enumeration class names should follow the same rules as class names (5.4.6), and should end with suffix "Kind". Note however that there are several base CIM classes that do not follow this naming convention.

5.15.5 Attribute

- Attribute names start with lower case (lowerCamelCase rule). Attribute names should use singular form. Attribute names are unique within a classifier. Inherited attribute names should be unique.

5.15.6 Enumeration literal

- Enumeration literal names should follow the same rules as attribute names (5.4.7), except in cases where the rules would change the meaning of the value.

5.15.7 Association

- Association names are not used and should be left empty.

5.15.8 Association end

- Association end names are mandatory at both sides for an association.
- Association end names start with upper case (UpperCamelCase rule).
- If association end has multiplicity greater than 1, it should still have the singular name. (Note that this historically has been inconsistently applied and we do not plan to change existing names just to meet this recommendation.)
- In case the role is clear from the referenced class the class name can be used as association end name. However, a role may have a specific meaning that does not correspond to the class name. In that case the meaning shall be captured in the association end name.
- In case there are multiple associations between two classes, it is essential to give different names to association ends of those associations. As a further generalisation of the previous sentence, it is also essential to give unique association end names for the opposite end class. Additionally, association end names should not duplicate attribute names on the opposite end class, though the likelihood of this is small given other naming conventions.
- Inherited association ends shall have unique names.

....

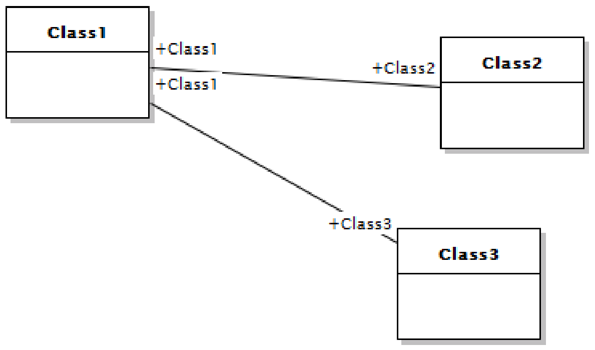


Figure 23 – Allowed duplication of association end names

Allowed duplication of association ends names is shown in Figure 23. Duplication of association end name “Class1” is allowed because one of them is seen by “Class2”, and the other is seen by “Class3”.

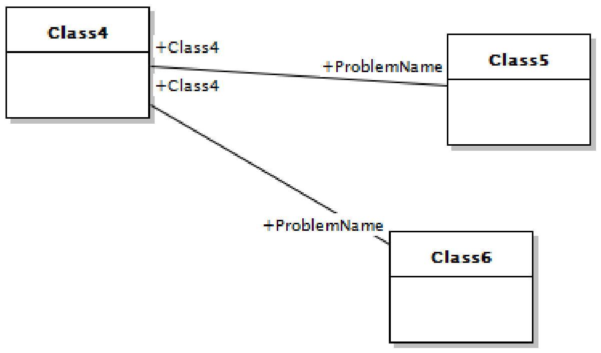


Figure 24 – Example of association end name duplication not allowed

Figure 24 shows a case where the association ends named “ProblemName” cause a problem. This case is not a valid CIM model, because “Class4” has two identical association end names and cannot distinguish which one is related to “Class5” and which one to “Class6”.

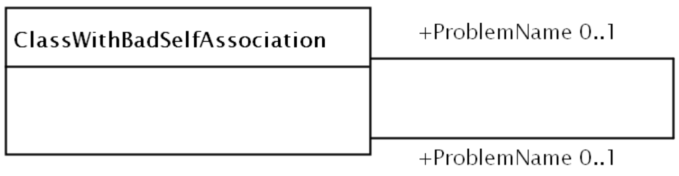


Figure 25 – Example of self-association end name duplication not allowed

A similar invalid case with of duplicate association end names using a self association is show in Figure 25.

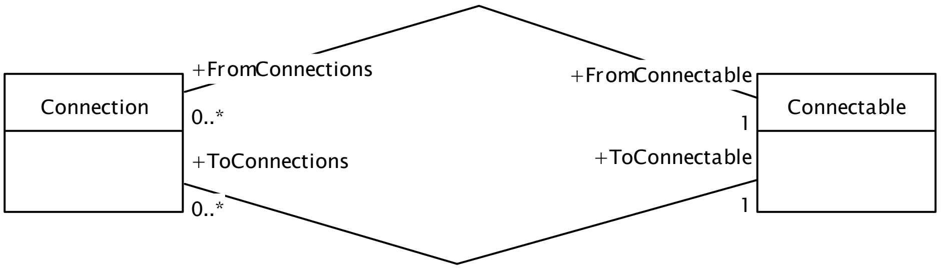


Figure 26 – Example of two associations among two classes

Multiple associations among two classes can be represented as in Figure 26. This example shows a Connection class that represents directed connections among two Connectable objects. In this example, Connectable object can have any number of connections.

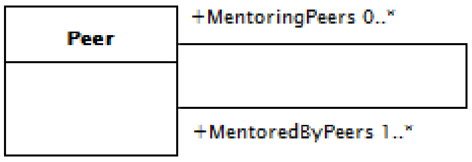


Figure 27 – Example association to self

A correct example of an association to self is given in Figure 27. This figure represents a group of peers where everyone may mentor other peers and everyone has at least one mentor that is a peer.

5.15.9 Diagram

- Diagram names start with upper case (UpperCamelCase rule). Diagram names shall be unique within the containing package. For example, there might be a “Main” diagram in more than one package, though it is encouraged using diagram names that are unique.

Rules related to profiling

6.1 General rules.

The following rules shall be respected:

f) A new class, attribute or association to be included in the IEC61970 or IEC61968 package are to be discussed within IEC TC 57 WG 13 and WG14 for approval. A proposal shall be made and the discussion is organised in the frame of a finalisation of a version of the canonical CIM.

g) A new class, attribute or association to be included in any of the CGMES related packages either in canonical CIM or in its extensions are to be discussed first within ENTSO-E CIM EG (CGMES SG) and then submitted to respective IEC TC 57 WG for approval.

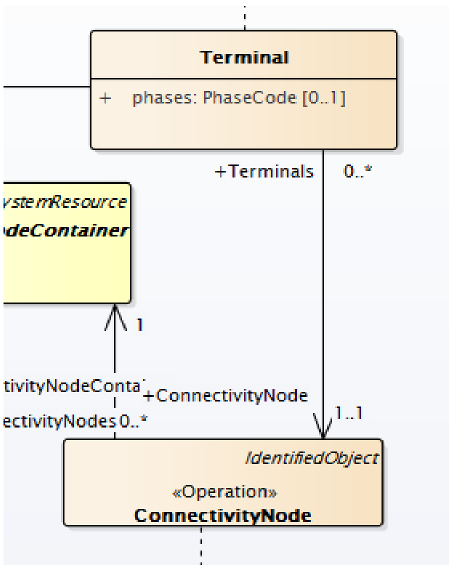
h) When designing a contextual document UML model (i.e. any of the profiles in UML), all classes, attributes and associations are to be “IsBasedOn” of a canonical CIM class, attribute or association.

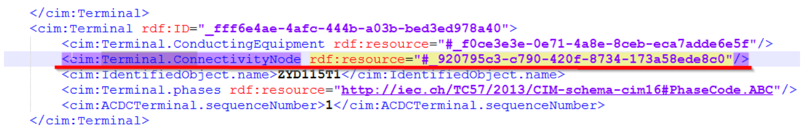
i) Depending upon the timelines of the standardisation process and a given business process, the corresponding UML package and related documentation shall either be submitted to IEC for standardization or remain at ENTSO-E level as “pre-standardisation” process. It is highly recommended that the objective of a particular work item proposal is clarified in the beginning of the process. If the objective is that particular work shall become as part of an international standard ENTSO-E working groups or project shall directly work on the preparation of the draft standards and make necessary consultations with relevant IEC working groups.

j) The multiplicity of attributes and associations' end roles shall not be less restrictive compared to the multiplicity defined in the canonical CIM (including extended part of the CIM). For instance a multiplicity can be set to 1..1 if in canonical CIM it is 0..1, but the other way around is not allowed.

k) The directions for all associations included in a profile shall be specified. This is specified by setting the navigability property option for association role ends. This is also independent to what is set as a source end and a target end for a given association. The direction is decided based on the use case, but it is recommended that the association points (is directed) to the end which is more restrictive multiplicity as follows (examples, possibly not a complete list of combinations):
    - Association 0..* - 1: the direction is at side with multiplicity 1
    - Association 0..* - 1..*: the direction is at side with multiplicity 1..*
    - Association 0..1 - 1: the direction is at side with multiplicity 1
    - Association 0..1 – 1..*: the direction is at side with multiplicity 1..*
    - Association 0..* - 0..1: the direction is at side with multiplicity 0..1
    - Association 0..* - 0..* (or 0..1 – 0..1): the direction is decided according to the use case.

The following figure illustrates the direction of the associations.





```
</cim:Terminal>
<cim:Terminal rdf:ID="_fff6e4ae-4afc-444b-a03b-bed3ed978a40">
    <cim:Terminal.ConductingEquipment rdf:resource="#_f0ce3e3e-0e71-4a8e-8ceb-eca7adde6e5f"/>
    <cim:Terminal.ConnectivityNode rdf:resource="#_920795c3-c790-420f-8734-173a58ede8c0"/>
    <cim:IdentifiedObject.name>ZYD115T1</cim:IdentifiedObject.name>
    <cim:Terminal.phases rdf:resource="http://iec.ch/TC57/2013/CIM-schema-cim16#PhaseCode.ABC"/>
    <cim:ACDCTerminal.sequenceNumber>1</cim:ACDCTerminal.sequenceNumber>
</cim:Terminal>
```

6.2 Rules about the organization of the profile packages.

The following rules shall be respected to support the maintenance of the EA file:

a) Profile packages when finalised shall not have substructure of packages. This is only allowed in transition period when modifications of a profile are proposed and not yet merged by CGMES SG with the main profile.

b) In cases where a new profile is to be developed, a new package shall be created in model in the area related to the profiles. As CGMES SG is responsible for the maintenance of the EA file the group shall be informed that a work for creation of a new profile is launched.

c) In cases where a modification of an existing profile is necessary by either adding new classes, attributes or associations or modifying existing elements the modifications need to be collected within a package under the profile package. CGMES SG shall be kept informed.

d) A profile package shall have at least one diagram created. It is highly recommended to use limited number of classes within a diagram and to have logical separation between what is to be presented in different diagrams. The meaning of the diagram shall be explained in its description.

e) In case a new profile package is created the package dependency diagram shown Figure 4 in need to be updated to include the newly created package and create the "IsBasedOn" dependency.

6.3 Rules on UML description

The following rules shall be respected regarding the descriptions of packages, classes, attributes (see also 5.25.10):

- In order to have a well documented UML package, it is necessary to provide the description of each profile package. This description is used when generating profile documentation.
- The descriptions of the classes and the attributes shall not be modified in the profiling step. In CGMES related profiles all descriptions are exactly the same as in the canonical CIM.
- By convention the descriptions of associations are not defined, i.e. only on the association end roles. Therefore, each association shall have a description of its end roles and these descriptions shall be the same as in the canonical CIM.

Step by step profiling using CimConteXtor

7.1 General overview

This guide does not focus on how CimConteXtor is used or launched as it is assumed that the reader is familiar with the manual of the add-in. This section aims at explaining the necessary steps to the profiling in a correct way and includes some hints on how complex profiling operations can be realised with CimConteXtor.

In order to cover the more complex profiling the section will focus on how a profile is created when the canonical CIM is extended. In cases where extension of the canonical CIM is not necessary the profiling is slightly different and this difference is indicated in the following sections.

7.2 Profile creation

There are basically two ways to create a profile using CimConteXtor. The manual method is where the user creates the profile package in the Enterprise Architect and then proceeds with adding necessary classes, attributes and associations. The other way is an automatic one where the utility CreateGlobalProfile is used (see CimConteXtor manual).

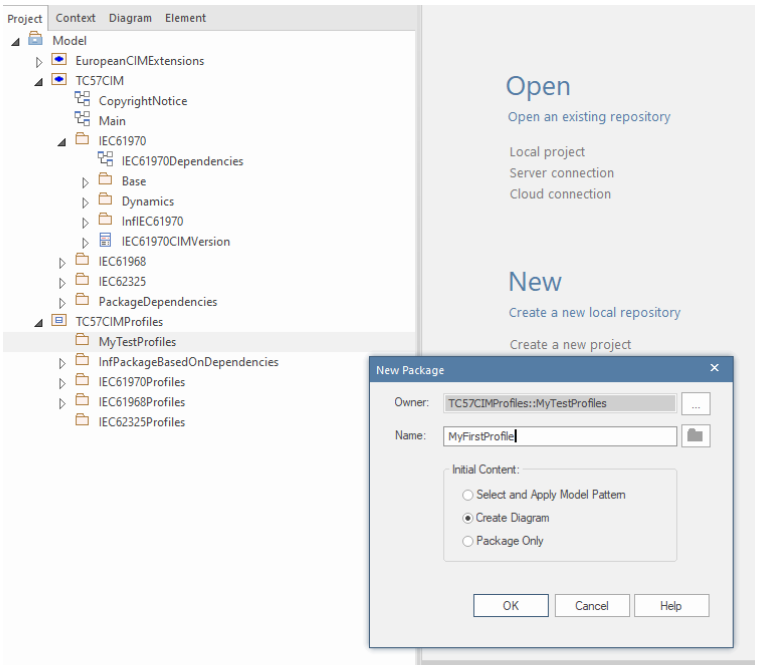
NOTE: Before starting the profiling work it is important to validate the canonical CIM (including extensions) using jCleanCim and fix all reported and critical issues. The report might contain known issues from the TC57CIM package and these shall be ignored for the profiling work in case they are known and not critical. However, the custom extensions shall be designed in a proper way hence no issues shall be found there.

7.2.1 Manual profile creation

In the manual process of profile creation, the following steps need to be followed:

7.2.1.1 Create the profile package in the hierarchy of profile packages

Using Enterprise Architect a new package is created (right click on the upper package and select “Add a Package”). In Figure 28 the package “MyTestProfiles” was cleated to contain a set of profiles. In the same figure it is seen that creation of the profile package “MyFirstProfile” is launched. For the proper work with CimConteXtor it is necessary to have at least one diagram under the profile package where the profiled classed will be visible. Therefore, it is useful to create this diagram at the time when the profile package is created (see Figure 28 where the option “Create Diagram” is selected). Note that the creation of the diagram can also be done afterwards.



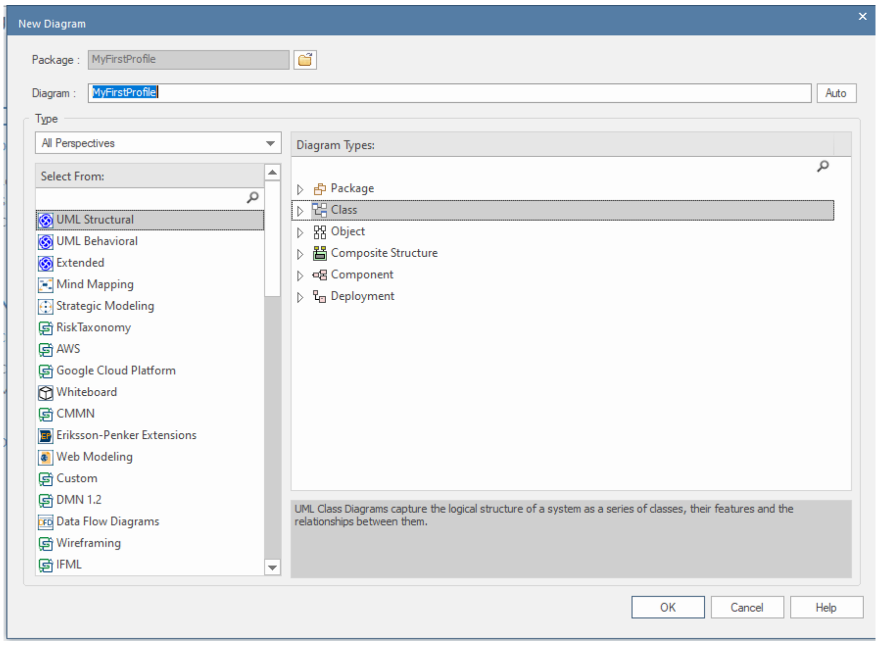
Owner: TC57CIMProfiles::MyTestProfiles

Name: MyFirstProfile

Initial Content:
- Select and Apply Model Pattern
- [✓ Create Diagram]
- Package Only

Figure 28 – Creation of the profile package

If it is selected to create a diagram the dialog window is shown in Figure 29, where the type of the diagram needs to be selected. Its type needs to be “Class” as the diagram will contain classes that are profiled.



Package : MyFirstProfile

Diagram : MyFirstProfile

Select From: UML Structural

Diagram Types: Class

UML Class Diagrams capture the logical structure of a system as a series of classes, their features and the relationships between them.

Figure 29 – Creation of diagram dialog window

The result is shown in Figure 30.

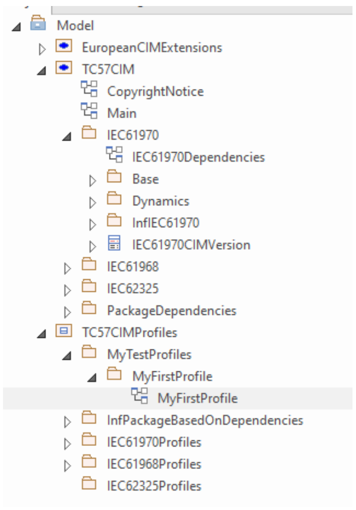


Figure 30 – Profile package created

7.2.1.2 Specify the package dependency.

In order to allow CimConteXtor to function properly it is necessary to define the dependency of the created profile package. The logic is that the profile depends on the canonical CIM. In the example it is assumed that this is a profile that uses classes from IEC61970 package and also uses classes from ExtIEC61970 package. In order to create the profile package dependency a package diagram shall be created. Then relevant packages shall be shown on the diagram (drag and drop the packages; select "Package element" when the context menu appears at the drop action). It is recommended that the package containing the diagram with the dependencies is included in the package that contains the profiles. The package name shall start with "Inf".

As in our case the profile will contain classes from IEC61970 and also from ExtIEC61970 packages the profile needs to have dependency with these packages. The dependencies are created using the arrow next to the class (see Figure 31). It is required to stereotype with "IsBasedOn" the dependencies of the profile package (e.g. MyFirstProfile) with the canonical CIM packages (e.g. ExtIEC61970 and IEC61970). It is also important to pay attention that dependencies between packages from canonical CIM (both from TC57CIM and extended packages) do not have "IsBasedOn".

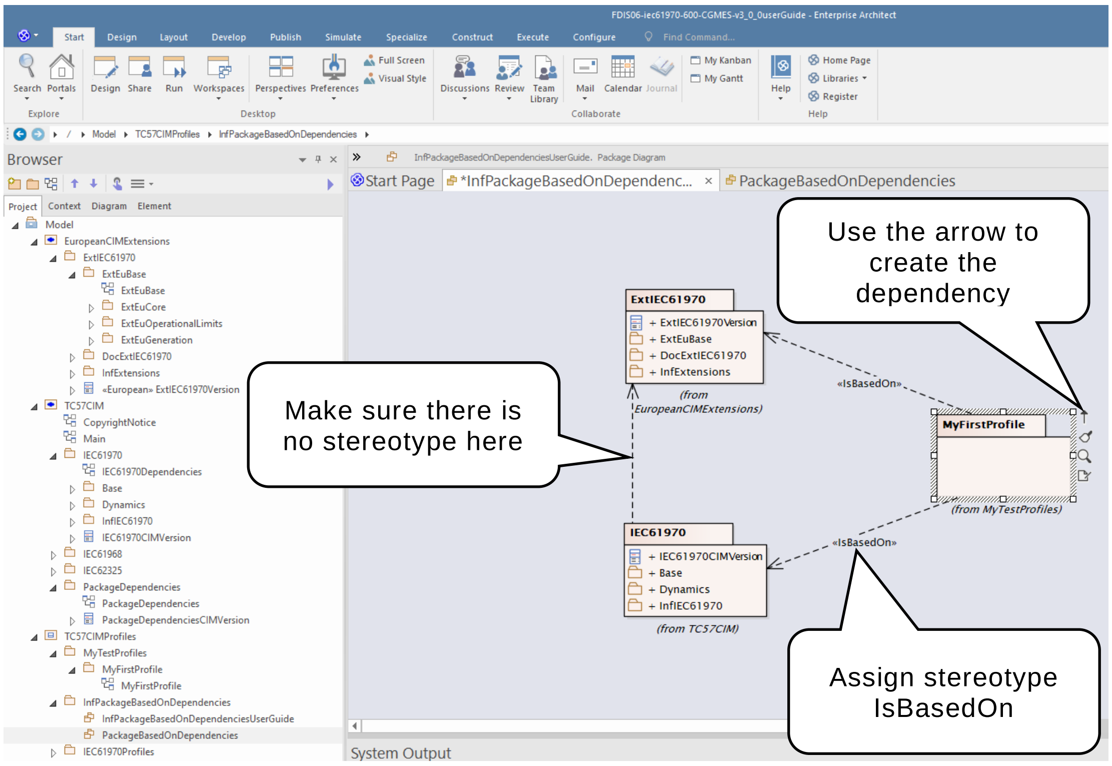
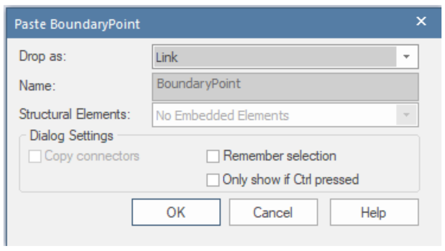


Figure 31 – Profile dependency

7.2.1.3 Adding a class to a profile

The way to add a class to a profile is to add the class to one of the diagrams in the profile. This operation is done with the support by CimConteXtor. The classical drag and drop function in Enterprise Architect is used to add a class to a diagram, however the following shall be taken into account:

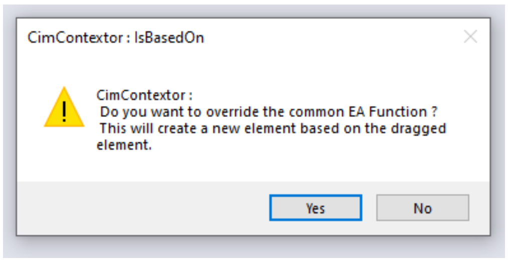
- The class shall be selected from a package part of canonical CIM (either TC57CIM or extension package). It is not allowed to select a class from another profile package.
- It is recommended that the profiling starts from the class in the upper hierarchy of the model that needs to be profiled. This will save time to complete the inheritance path.
- NOTE: It is recommended that before starting with drag and drop the diagram where the class shall be dropped is saved and there are only a few diagrams opened. Preferably only one diagram. In case the diagram is not saved the arrangements done before drag and drop might be lost. In case there are too many diagrams opened CimConteXtor will reopen and save all of them during the profiling step, which takes time.
- The dependency between the profile package and the canonical CIM package shall be defined beforehand (see 7.2.1.2)
- The class shall be dropped "as Link" as shown in Figure 2Figure 32. "OK" button shall be selected.


Paste BoundaryPoint

Drop as: Link
Name: BoundaryPoint
Structural Elements: No Embedded Elements
Dialog Settings
- [ ] Copy connectors
- [ ] Remember selection
- [ ] Only show if Ctrl pressed

Figure 32 – Drop a class in a profile

- The dialog by CimConteXtor will follow as shown in Figure 33. "Yes" button shall be selected, which will start the process to create the class in the profile. Note that if by a mistake you click "No", the created class in the diagram will be just a link to the canonical CIM class so it will not be profiled, i.e. it will not appear in the profile package, but it will only be in the diagram. In case, you see that the class is still created in the profile package you need to go to the project browser of Enterprise Architect and delete the class before repeating the operation again. It is not enough to delete the class from the diagram as this will only delete the geographical representation of the class in the diagram and not the class in the profile.


CimContextor : IsBasedOn

CimContextor :
Do you want to override the common EA Function ?
This will create a new element based on the dragged element.

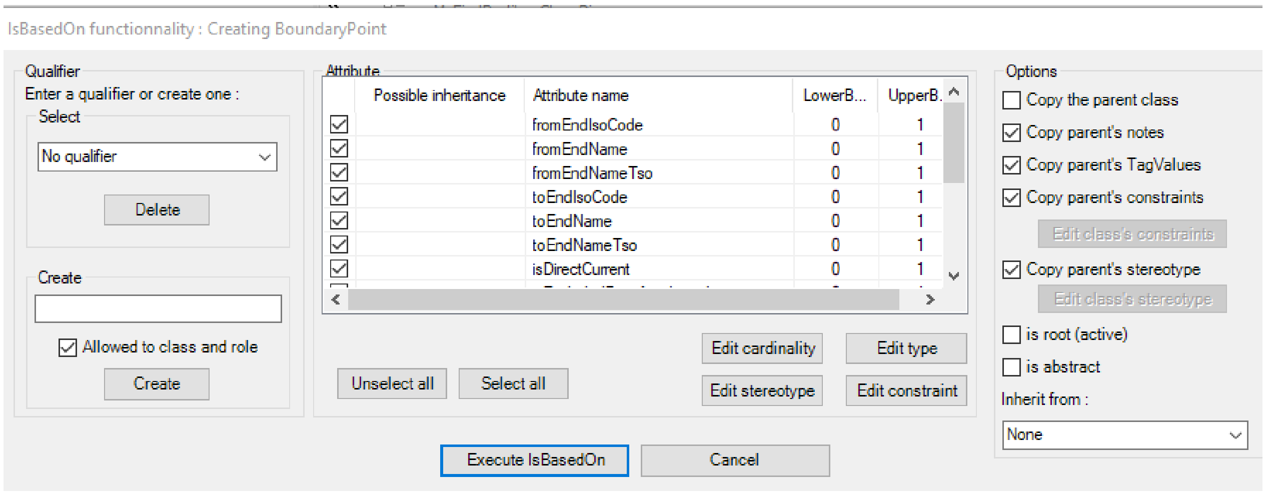
Figure 33 – CimConteXtor "IsBasedOn" dialog

- The dialog on Figure 34 will open. The details on what can be done in this dialog are explained in the CimConteXtor manual. Here another technique will be explained that speeds up the profiling process. There are two important things that need to be selected in this dialog as follows:
    - The property "Inherit from" (Figure 34). This property shall be selected first in order to see which attributes are already inherited and they shall not be selected. This is why it is important that classes that are upper in the hierarchy of the model are already in the profile. If this is not the case nothing could be selected in the property "Inherit from" and basically this action shall be completed by using "Edit IsBasedOn" feature

at later stage. This is possible but more time consuming. Therefore, the person that profiles shall make a plan from where to start the profile.

- The property "Attribute". The best is to complete this once the inheritance is selected so that you have an indication which attributes are inherited and shall not be selected as attributes of the class that is to be profiled. NOTE: The attributes that have the symbol "*" in the column "Possible inheritance" (Figure 34) shall not be selected. This will create that attribute on two places which is not allowed. For example, if the class inherits from IdentifiedObject, the attribute .name shall not be selected as attribute also for the profiled class.

When completing these actions the button "Execute IsBasedOn" shall be selected. Other properties can be adjusted in the properties of the class and attributes that Enterprise Architect provides.


IsBasedOn functionnality : Creating BoundaryPoint

Qualifier
Enter a qualifier or create one :
Select: No qualifier
Delete
Create
☑ Allowed to class and role
Create

Attribute

	Possible inheritance	Attribute name	LowerB...	UpperB...
☑		fromEndIsoCode	0	1
☑		fromEndName	0	1
☑		fromEndNameTso	0	1
☑		toEndIsoCode	0	1
☑		toEndName	0	1
☑		toEndNameTso	0	1
☑		isDirectCurrent	0	1

Unselect all · Select all · Edit cardinality · Edit type · Edit stereotype · Edit constraint

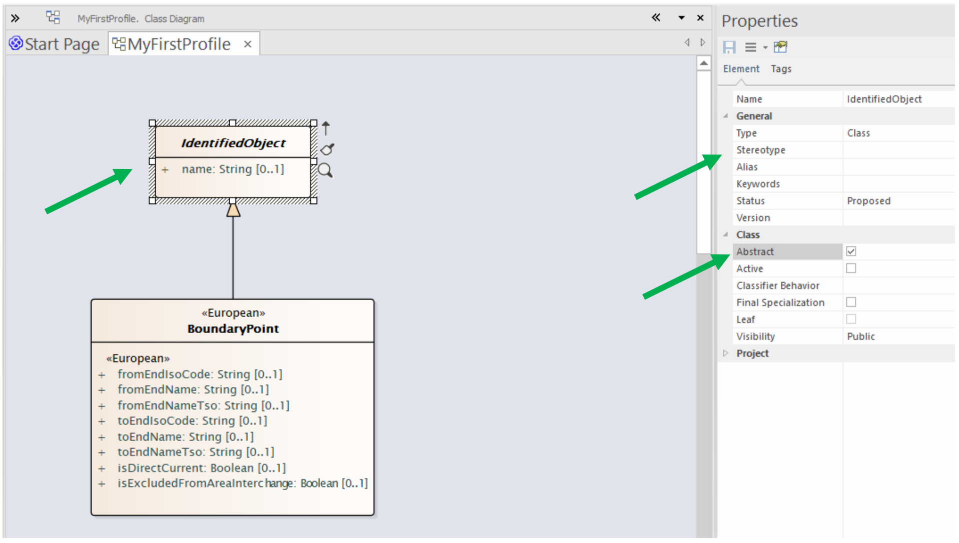
Options
☐ Copy the parent class
☑ Copy parent's notes
☑ Copy parent's TagValues
☑ Copy parent's constraints
Edit class's constraints
☑ Copy parent's stereotype
Edit class's stereotype
☐ is root (active)
☐ is abstract
Inherit from : None

Execute IsBasedOn · Cancel

Figure 34 – CimConteXtor creating class dialog

7.2.1.4 Adjust additional properties of a profiled class using Enterprise Architect

A class which is already profiled can be adjusted directly using properties of Enterprise Architect as shown in Figure 35.


MyFirstProfile. Class Diagram
Start Page
MyFirstProfile
IdentifiedObject
+ name: String [0..1]
«European»
BoundaryPoint
«European»
+ fromEndIsoCode: String [0..1]
+ fromEndName: String [0..1]
+ fromEndNameTso: String [0..1]
+ toEndIsoCode: String [0..1]
+ toEndName: String [0..1]
+ toEndNameTso: String [0..1]
+ isDirectCurrent: Boolean [0..1]
+ isExcludedFromAreaInterchange: Boolean [0..1]
Properties
Element
Tags

Name	IdentifiedObject
General	
Type	Class
Stereotype	
Alias	
Keywords	
Status	Proposed
Version	
Class	
Abstract	☑
Active	☐
Classifier Behavior	
Final Specialization	☐
Leaf	☐
Visibility	Public
Project	

Figure 35 – Adjust class properties

It is very convenient if the Enterprise Architect is setup in a way that the "Properties" are directly shown[2] as seen in Figure 35. This allows direct editing without launching other dialogs, i.e. the modifications are fast. In most cases the stereotype and the abstract properties are to be controlled

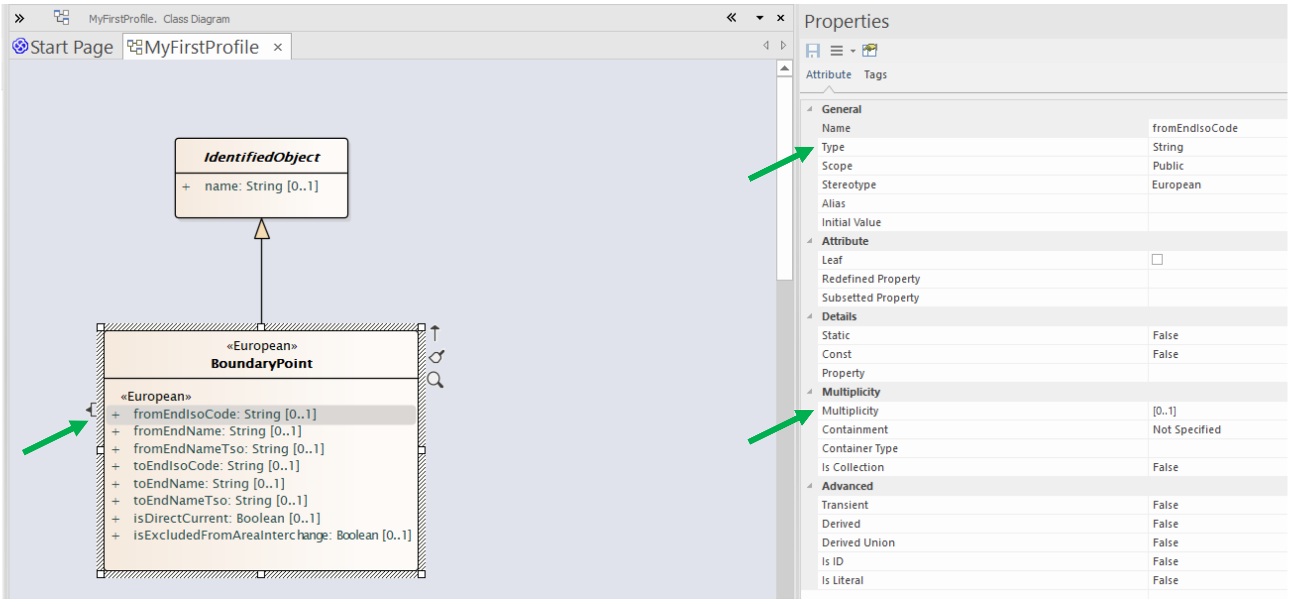
[2] Different versions of Enterprise Architect may have different ways how to show the Properties. In some versions right click of the class and selecting show tagged values can be used.

and adjusted if necessary. In the current CGMES profiling techniques there is only one additional stereotype that can be specified at the profiling stage. This is the stereotype “Description” which is assigned for classes included in profiles that complete the class, e.g. Steady State Hypothesis profile mostly include classes stereotyped with “Description” because these classes just add elements to classes that first instantiated in Equipment profile. This approach is kind of temporary as moving forward it is expected that there will be an adjustment in the serialisation to only use rdf:about, hence this stereotype will not be needed anymore.

The class shall be made abstract in the profile in cases where an instance of this class in not expected to be exchanged. In most cases a class is set as abstract where the class has subclasses. However, this depends on the profile and what needs to be modelled.

7.2.1.5 Adjust additional properties of attributes using Enterprise Architect

The attributes of a class which is already profiled can be adjusted directly using properties of Enterprise Architect as shown in Figure 36.


MyFirstProfile. Class Diagram
Start Page
MyFirstProfile
IdentifiedObject
+ name: String [0..1]
«European»
BoundaryPoint
«European»
+ fromEndIsoCode: String [0..1]
+ fromEndName: String [0..1]
+ fromEndNameTso: String [0..1]
+ toEndIsoCode: String [0..1]
+ toEndName: String [0..1]
+ toEndNameTso: String [0..1]
+ isDirectCurrent: Boolean [0..1]
+ isExcludedFromAreaInterchange: Boolean [0..1]
Properties
Attribute
Tags

General	
Name	fromEndIsoCode
Type	String
Scope	Public
Stereotype	European
Alias	
Initial Value	
Attribute	
Leaf	☐
Redefined Property	
Subsetted Property	
Details	
Static	False
Const	False
Property	
Multiplicity	
Multiplicity	[0..1]
Containment	Not Specified
Container Type	
Is Collection	False
Advanced	
Transient	False
Derived	False
Derived Union	False
Is ID	False
Is Literal	False

Figure 36 – Adjust attribute properties

With properties shown as in Figure 36 it is sufficient to just select the attribute that needs to be modified. The most important property to adjust is the multiplicity as this can be further restricted compared to the multiplicity in the canonical CIM. It is not allowed to relax the multiplicity in the profiling step in cases where stricter multiplicity is defined in the canonical CIM.

The type property also needs to be changed to be make sure that the type is local in the profile. Please note that before this action is applied all used types in the profile shall already be profiled. Two approaches can be applied when changing the types to be local:

- In cases of very small profiles or just single modifications the properties item can be used and select the right datatype which is local in the profile.
- Use the utility “RecoverProfileDatatypes” (the way to launch it is shown in Figure 37). This is the recommended approach and normally this utility shall be run as one of the last steps of the profiling routine in order to ensure that all datypes are local in the profile.

  Note that the prerequisite is that all datatypes are already in the profile. If this is not the case the utility “LocalizeDataTypes” can be used, but still in order to properly execute this utility there is a need to have a profiled datatypes as part of the EA file.

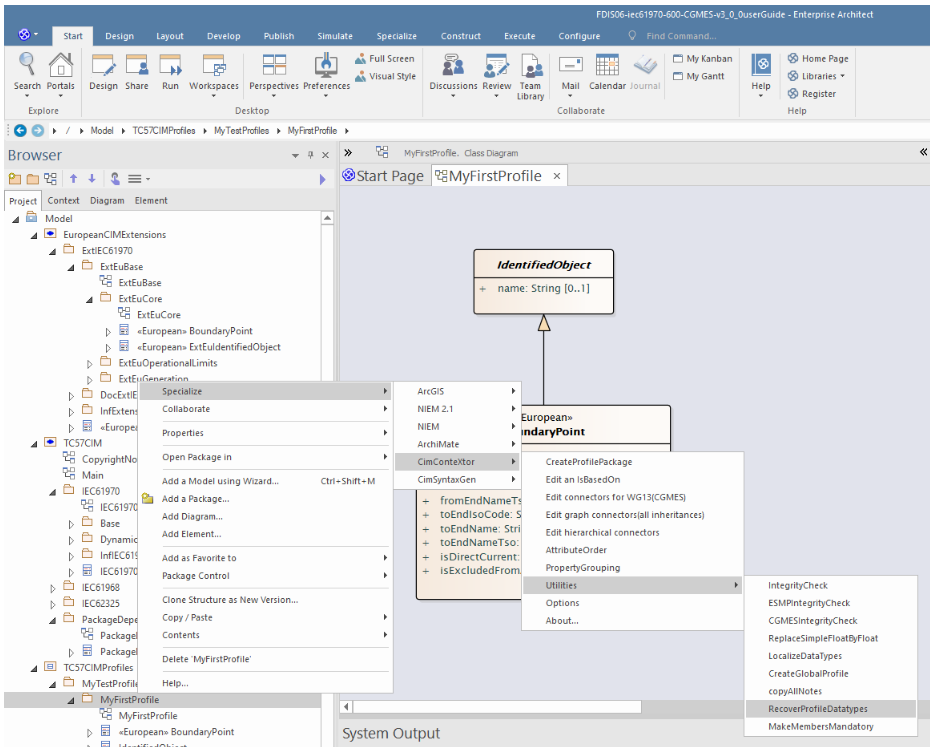


Figure 37 – Utility “RecoverProfileDatatypes”

7.2.1.6 Using “Edit IsBasedOn” by CimConteXtor

To change any characteristic of a class or an attribute, the option “Edit IsBasedOn” by CimConteXtor can be used. In order to do this, select the class in the UML class diagram, right click to open the CimConteXtor menu and select the “Edit IsBasedOn” as shown in Figure 38.

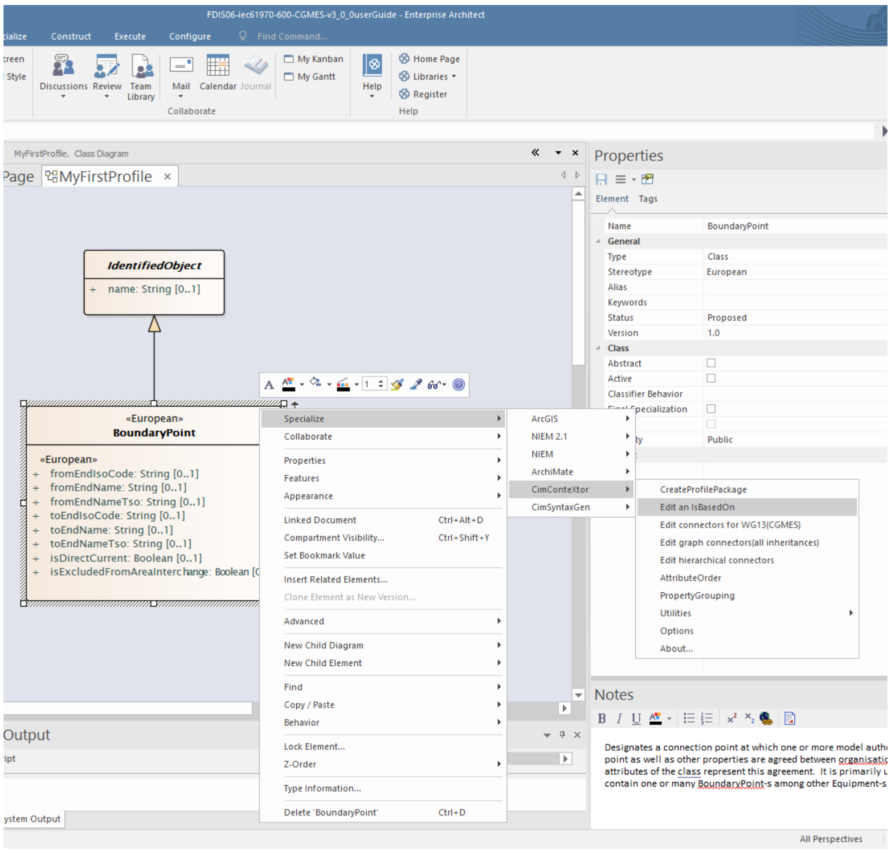


Figure 38 – CimConteXtor “Edit IsBasedOn” launch

7.2.1.7 Adding an association to a profile

Before adding an association to a profile the two classes between which the association exists shall already be profiled. For example, in Figure 39 the classes BoundaryPoint and ConnectivityNode are already added in the profile. The task is to profile the association between the two classes.

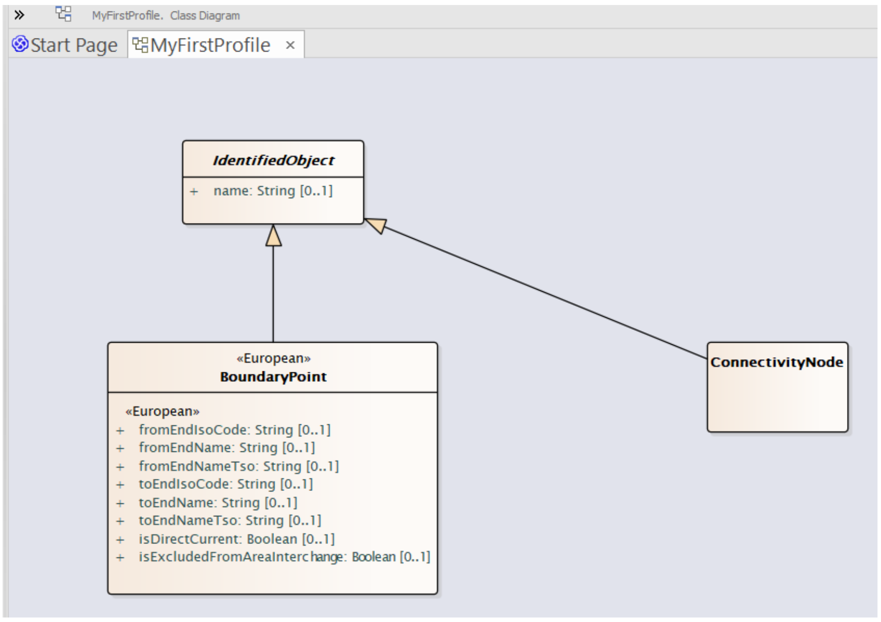

MyFirstProfile. Class Diagram
Start Page
MyFirstProfile
IdentifiedObject
+ name: String [0..1]
«European»
BoundaryPoint
«European»
+ fromEndIsoCode: String [0..1]
+ fromEndName: String [0..1]
+ fromEndNameTso: String [0..1]
+ toEndIsoCode: String [0..1]
+ toEndName: String [0..1]
+ toEndNameTso: String [0..1]
+ isDirectCurrent: Boolean [0..1]
+ isExcludedFromAreaInterchange: Boolean [0..1]
ConnectivityNode

Figure 39 – Preparation before adding an association

Depending on where in the model hierarchy the class to which the association shall be added is positioned the task to add an association could be time consuming. CimConteXtor was optimised to cope with this, but still it is recommended to select the class that presumably has less connections in the model (including inherited). In the example of Figure 39 the class BoundaryPoint is an extension and it is better to execute the procedure of adding an association on this class, in this way the procedure will be quicker. The way to launch the adding of an association is shown in Figure 40.

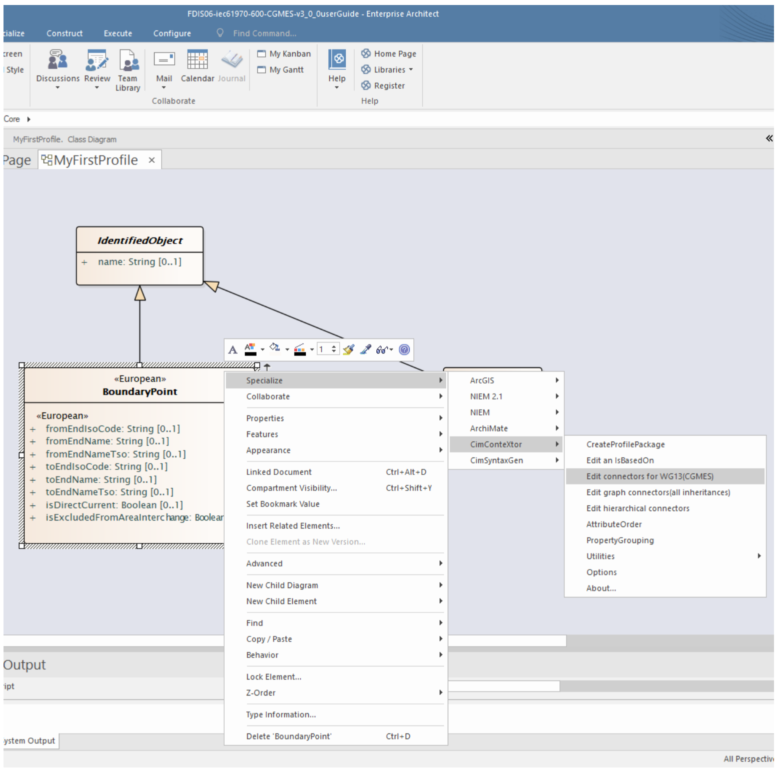
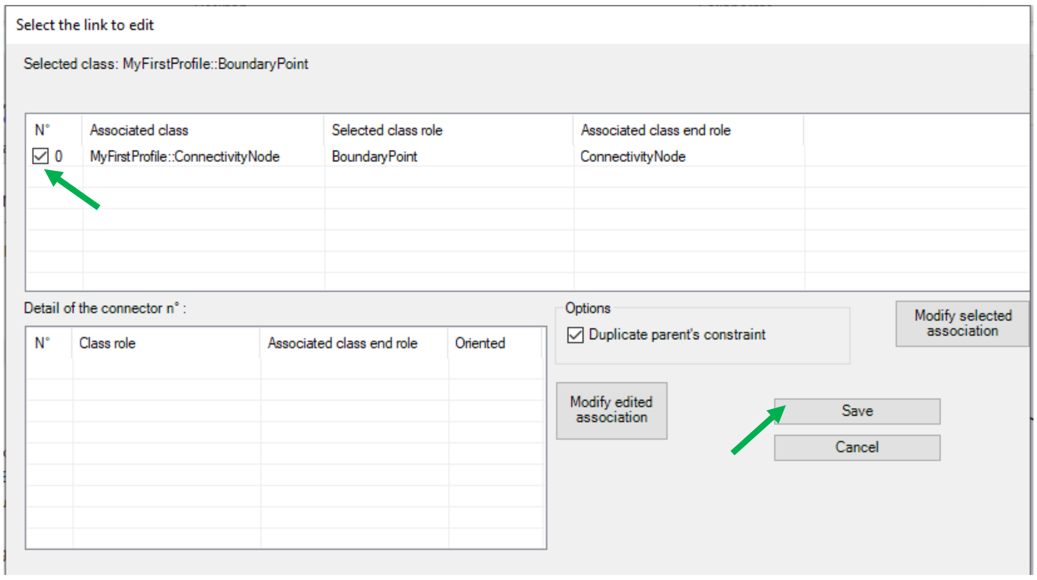


Figure 40 – Using CimConteXtor to add an association

“Edit connections for WG13 (CGMES)” shall be selected. The dialog shown in Figure 41 appears. In this dialog a lost with existing associations for BoundaryPoint class are shown. The desired association shall be selected be checking the checkbox in the beginning of the association row as shown in Figure 41.



Select the link to edit

Selected class: MyFirstProfile::BoundaryPoint

N°	Associated class	Selected class role	Associated class end role
☑ 0	MyFirstProfile::ConnectivityNode	BoundaryPoint	ConnectivityNode

Detail of the connector n° :

N°	Class role	Associated class end role	Oriented

Options

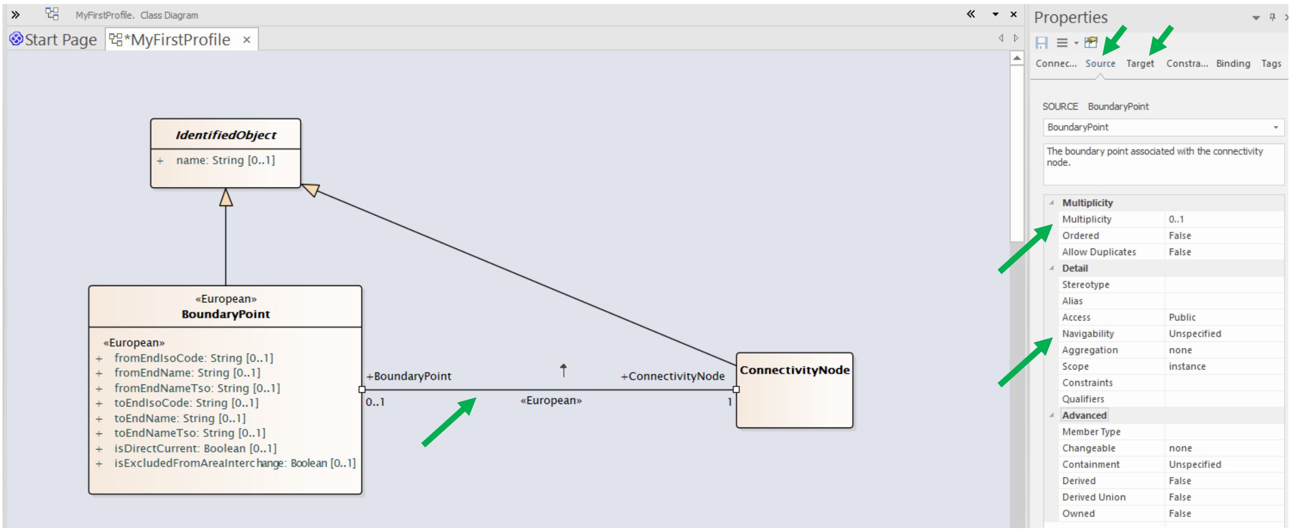
☑ Duplicate parent's constraint

Figure 41 – Edit association dialog

“Save” button is selected to add the association. CimConteXtor offers additional dialogs that can edit and further specify the properties of the association before it is added. This is explained in CimConteXtor manual. However, this task might be time consuming, hence this user guide explains how the further adjustments of the properties can be done by the Enterprise Architect dialogs (see 7.2.1.8).

7.2.1.8 Adjusting properties of an association

An association which is already profiled can be adjusted directly using properties of Enterprise Architect as shown in Figure 42. When the association is selected the properties for the “Source” and the “Target” roles of the association are visible and can be edited. Note that the Enterprise Architect shall be set to show the properties in the way they are shown in Figure 42.



MyFirstProfile. Class Diagram
Start Page
*MyFirstProfile
IdentifiedObject
+ name: String [0..1]
«European»
BoundaryPoint
«European»
+ fromEndIsoCode: String [0..1]
+ fromEndName: String [0..1]
+ fromEndNameTso: String [0..1]
+ toEndIsoCode: String [0..1]
+ toEndName: String [0..1]
+ toEndNameTso: String [0..1]
+ isDirectCurrent: Boolean [0..1]
+ isExcludedFromAreaInterchange: Boolean [0..1]
+BoundaryPoint
0..1
«European»
+ConnectivityNode
1
ConnectivityNode
Properties
Connec...
Source
Target
Constra...
Binding
Tags

SOURCE BoundaryPoint

BoundaryPoint

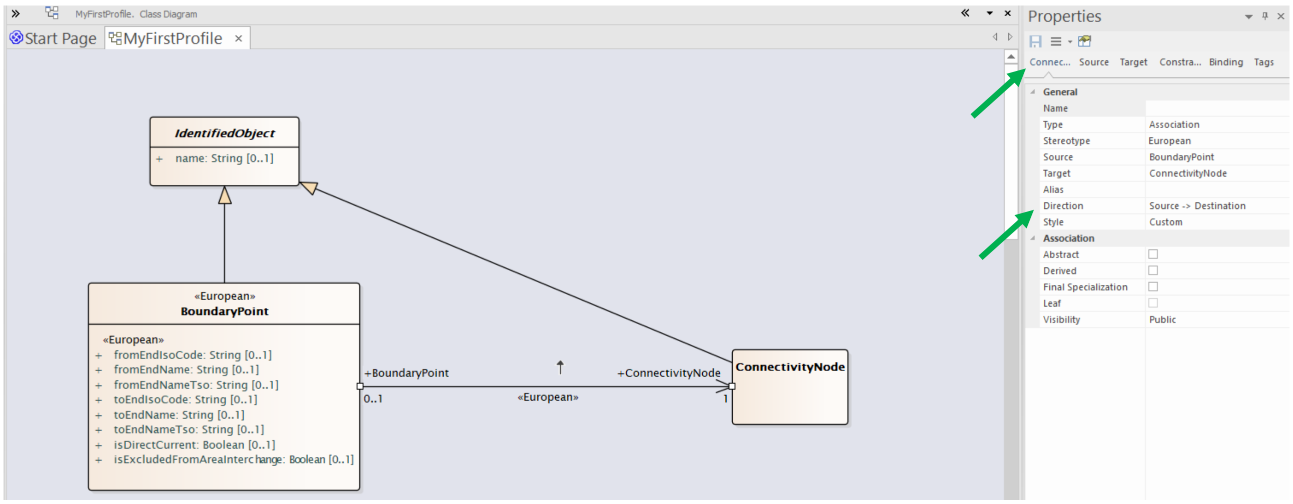
The boundary point associated with the connectivity node.

Multiplicity	
Multiplicity	0..1
Ordered	False
Allow Duplicates	False
Detail	
Stereotype	
Alias	
Access	Public
Navigability	Unspecified
Aggregation	none
Scope	instance
Constraints	
Qualifiers	
Advanced	
Member Type	
Changeable	none
Containment	Unspecified
Derived	False
Derived Union	False
Owned	False

Figure 42 – Edit properties of an association in Enterprise Architect

There are two important properties that need to be considered when editing the profiled association. These are the “Multiplicity” and “Navigability” of the two ends of the association.

The multiplicity of an association is only changed if in the profile this association shall be further restricted. It is not allowed that the multiplicity is less restrictive compared to the multiplicity defined in the canonical CIM (including extended canonical part or CIM).
NOTE: The direction of the associations shall be specified in the profile for all associations (see rules in 6.1). It is important to note that CimConteXtor considers that the source of the association is the class on which the procedure to add the association in the profile was launched. Therefore, this shall be taken into account if the direction of the association is set by using the “Direction” property (and not the navigability option at the role ends) in the Enterprise Architect as shown in Figure 43. In the example case it is selected “Source->Destination” because the association was added on the class BoundaryPoint. If the adding of the association was on ConnectivityNode, the proper direction would be given if the property is set to “Destination->Source”.


MyFirstProfile. Class Diagram
Start Page
MyFirstProfile
IdentifiedObject
+ name: String [0..1]
«European»
BoundaryPoint
«European»
+ fromEndIsoCode: String [0..1]
+ fromEndName: String [0..1]
+ fromEndNameTso: String [0..1]
+ toEndIsoCode: String [0..1]
+ toEndName: String [0..1]
+ toEndNameTso: String [0..1]
+ isDirectCurrent: Boolean [0..1]
+ isExcludedFromAreaInterchange: Boolean [0..1]
+BoundaryPoint
0..1
«European»
+ConnectivityNode
1
ConnectivityNode
Properties
Connec...
Source
Target
Constra...
Binding
Tags

General	
Name	
Type	Association
Stereotype	European
Source	BoundaryPoint
Target	ConnectivityNode
Alias	
Direction	Source -> Destination
Style	Custom
Association	
Abstract	☐
Derived	☐
Final Specialization	☐
Leaf	☐
Visibility	Public

Figure 43 – Editing the direction of an association

The result of a profiled association with navigability set is shown in Figure 44.

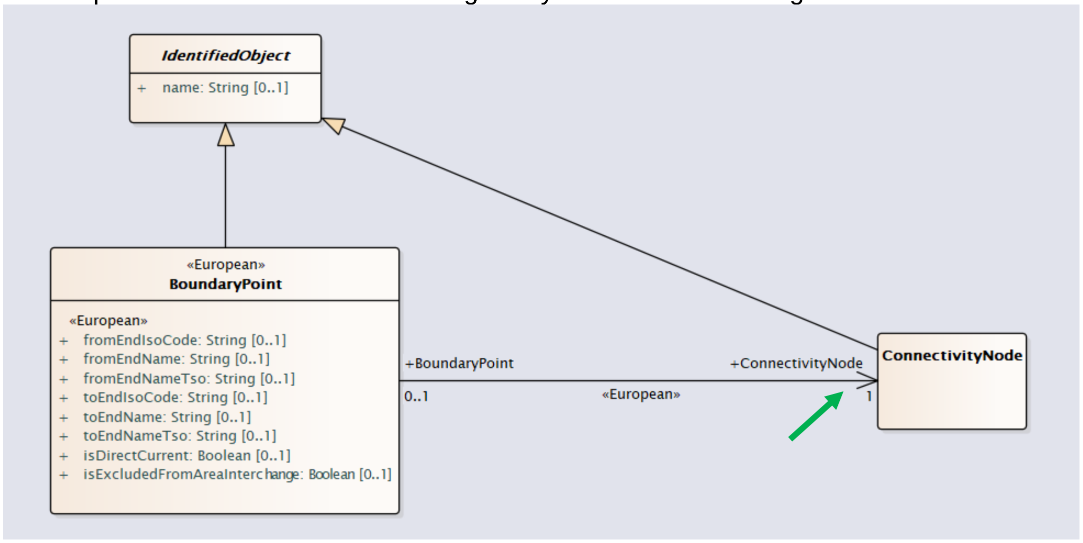

IdentifiedObject
+ name: String [0..1]
«European»
BoundaryPoint
«European»
+ fromEndIsoCode: String [0..1]
+ fromEndName: String [0..1]
+ fromEndNameTso: String [0..1]
+ toEndIsoCode: String [0..1]
+ toEndName: String [0..1]
+ toEndNameTso: String [0..1]
+ isDirectCurrent: Boolean [0..1]
+ isExcludedFromAreaInterchange: Boolean [0..1]
+BoundaryPoint
0..1
«European»
+ConnectivityNode
1
ConnectivityNode

Figure 44 – Profiled association

7.2.2 Semi-automatic profile creation

The utility CreateGlobalProfile can be found as shown in Figure 45. When using this utility, it is important to know that:

- The utility needs to be launched on the package which is part of the canonical CIM (either from the extended part or from TC57 part of canonical CIM).
- The utility is mostly applied in situations where all elements part of a package shall be profiled in one profile. Therefore, it is not useful to profile Base package from TC57CIM->IEC61970 package as the content of this package is part of different profiles, however it was used, for instance, to

profile Dynamics package from TC57CIM->IEC61970 package where all elements are part of the IEC 61970-457 profile.

- The utility helps saving time to profile many classes, however it cannot be expected that the profile is complete after the execution of the utility. Still restrictions on attributes and associations shall be applied. In this respect, please note that another utility MakeMembersMandatory (see Figure 45) can be used to support the used to change the multiplicity of all attributes of classes in a given package to 1..1.
- The utility can be used as a part of more complex profiling where parts of the profile can be generated in the automatic way and parts in manual (classical) way. This can be realised by grouping classes that needs to be profiled in one (temporary) package in the canonical part of CIM and then this package is profiled. Such technique needs to be used on a separate copy of the EA file as the canonical CIM will need to be modified temporary in order to achieve the profiling task quicker. The resulting profile then will need to be exported and combined with the original canonical CIM so that in the EA file where the profile and the canonical CIM are maintained the canonical CIM is the same as in the original and not as in the changed temporally way of grouping of classes.

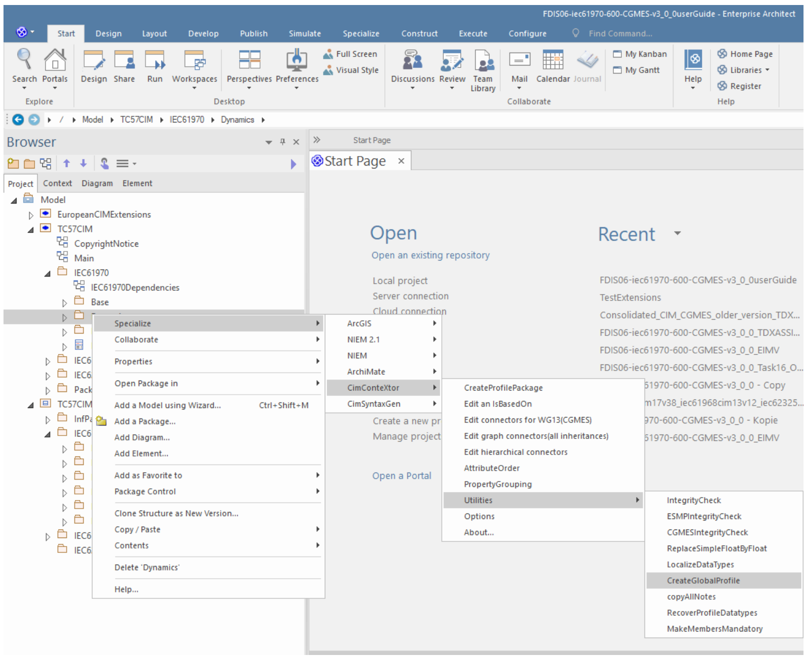


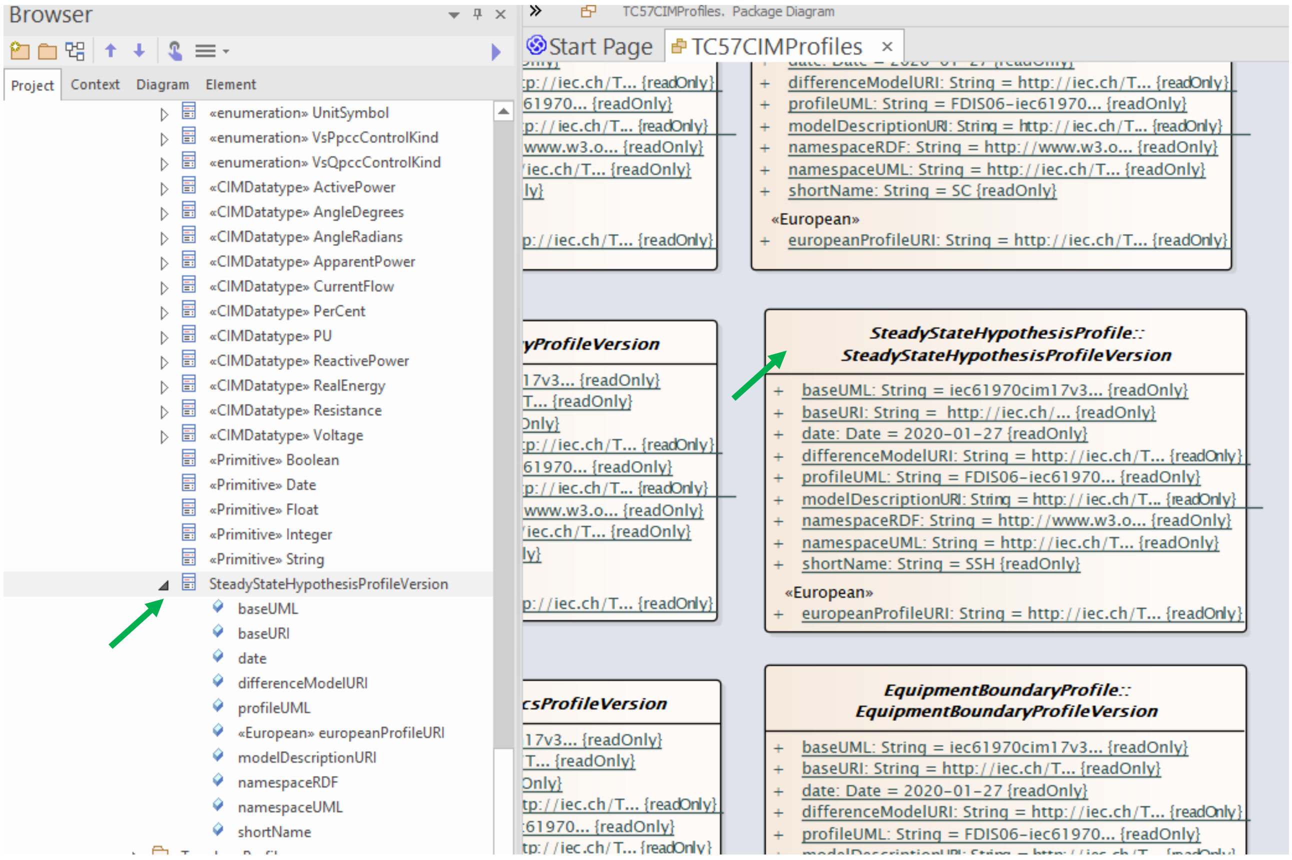
Figure 45 – Launching of CreateGlobalProfile utility

7.2.3 Finalization of the profile

There are for activities that shall be performed at the final stage before the profile is distributed for standardisation, review or implementation. The tasks are as follows:

- Create version class with necessary attributes.
- Run utility “RecoverProfileDatatypes” (see 8.4)
- Run utility “CopyAllNotes” (see 8.5)
- Run utility “CGMESIntegrityCheck” (see 8.3)
- Compact EA file (see 8.6)

Each profile shall have a class that specifies information related to the versioning of the profile. In earlier CGMES versions a version class in the profile is used by CimConteXtor utilities (e.g. “CGMESIntegrityCheck”), thus the content of this class shall be according to the example as shown in Figure 46.



«enumeration» UnitSymbol
«enumeration» VsPpccControlKind
«enumeration» VsQpccControlKind
«CIMDatatype» ActivePower
«CIMDatatype» AngleDegrees
«CIMDatatype» AngleRadians
«CIMDatatype» ApparentPower
«CIMDatatype» CurrentFlow
«CIMDatatype» PerCent
«CIMDatatype» PU
«CIMDatatype» ReactivePower
«CIMDatatype» RealEnergy
«CIMDatatype» Resistance
«CIMDatatype» Voltage
«Primitive» Boolean
«Primitive» Date
«Primitive» Float
«Primitive» Integer
«Primitive» String
SteadyStateHypothesisProfileVersion
baseUML
baseURI
date
differenceModelURI
profileUML
«European» europeanProfileURI
modelDescriptionURI
namespaceRDF
namespaceUML
shortName

```
+ differenceModelURI: String = http://iec.ch/T... {readOnly}
+ profileUML: String = FDIS06-iec61970... {readOnly}
+ modelDescriptionURI: String = http://iec.ch/T... {readOnly}
+ namespaceRDF: String = http://www.w3.o... {readOnly}
+ namespaceUML: String = http://iec.ch/T... {readOnly}
+ shortName: String = SC {readOnly}
«European»
+ europeanProfileURI: String = http://iec.ch/T... {readOnly}
```

```
SteadyStateHypothesisProfile::
SteadyStateHypothesisProfileVersion
+ baseUML: String = iec61970cim17v3... {readOnly}
+ baseURI: String = http://iec.ch/... {readOnly}
+ date: Date = 2020-01-27 {readOnly}
+ differenceModelURI: String = http://iec.ch/T... {readOnly}
+ profileUML: String = FDIS06-iec61970... {readOnly}
+ modelDescriptionURI: String = http://iec.ch/T... {readOnly}
+ namespaceRDF: String = http://www.w3.o... {readOnly}
+ namespaceUML: String = http://iec.ch/T... {readOnly}
+ shortName: String = SSH {readOnly}
«European»
+ europeanProfileURI: String = http://iec.ch/T... {readOnly}
```

```
EquipmentBoundaryProfile::
EquipmentBoundaryProfileVersion
+ baseUML: String = iec61970cim17v3... {readOnly}
+ baseURI: String = http://iec.ch/T... {readOnly}
+ date: Date = 2020-01-27 {readOnly}
+ differenceModelURI: String = http://iec.ch/T... {readOnly}
+ profileUML: String = FDIS06-iec61970... {readOnly}
```

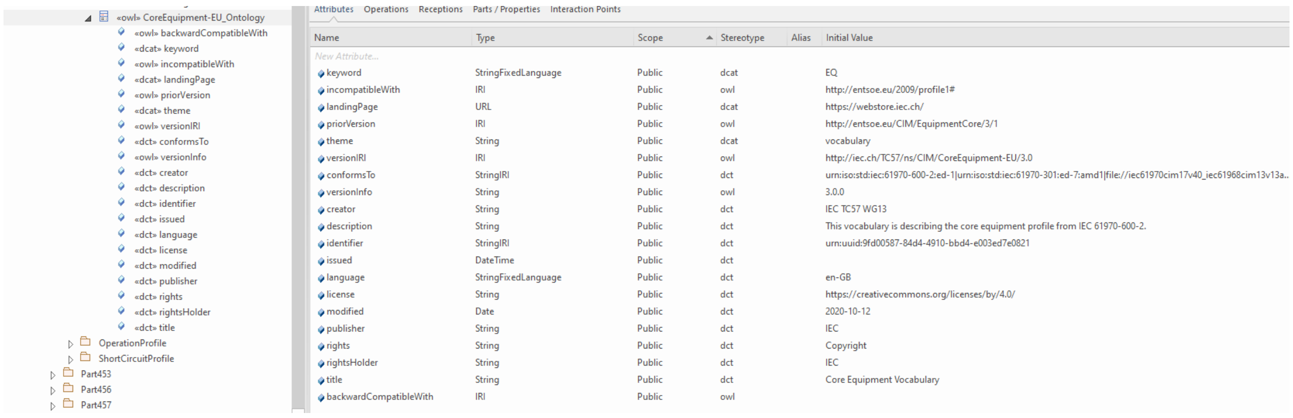
Figure 46 – Example of a version class in a profile

Please note that the name of the class shall be the name of the profile + 'Version'. In the example in Figure 46 the name is SteadyStateHypothesisProfileVersion which is something that CimConteXtor will not find and the integrity check will not work. The proper class name in that case is SteadyStateHypothesisVersion.

In the CGMES v3 version class was replaced with an ontology class. This approach should be used when creating new profiles. CimSyntaxGen is using this ontology class in order to populate the file header properties of the RDFS and SHACL exports. The ontology class is designed to be in line with W3C recommendations and uses the definitions specified in IEC 61970-501:Ed2[3]. Figure 47 illustrates the ontology class for Core Equipment profile in CGMES v3. The class is stereotypes with "owl" it is an abstract class and has initial values for relevant attributes. Please note that the attribute "issued" has no initial value defined because this is autogenerated at the time of the export of the related artefact. The name of the class is composed of the name of the profile as a qualifier and "Ontology". This class is created in InfConfiguration package which contacts all the meta information related to profiles. However, the class in then moved to the related profile package.
Classes in InfConfiguration package are profiled using CimConteXtor and are based on the IEC 61970-501 profile. The classes in InfConfiguration are not representing a profile, but they are kind of instances of the IEC 61970-501 profile which is describing the relationship between different artifacts (e.g. vocabulary – an RDF serialisation of the schema – RDF XML; constraints – RDF serialisation of SHACL Shapes) of a one or multiple profiles.

Figure 48 illustrates the InfConfiguration for Core Equipment profile in CGMES v3. Please note that this approach might be further developed in the process of finalising IEC 61970-501.

[3] The IEC 61970-501 is at CD stage.



Name	Type	Scope	Stereotype	Alias	Initial Value
keyword	StringFixedLanguage	Public	dcat		EQ
incompatibleWith	IRI	Public	owl		http://entsoe.eu/2009/profile1#
landingPage	URL	Public	dcat		https://webstore.iec.ch/
priorVersion	IRI	Public	owl		http://entsoe.eu/CIM/EquipmentCore/3/1
theme	String	Public	dcat		vocabulary
versionIRI	IRI	Public	owl		http://iec.ch/TC57/ns/CIM/CoreEquipment-EU/3.0
conformsTo	StringIRI	Public	dct		urn:iso:std:iec:61970-600-2:ed-1|urn:iso:std:iec:61970-301:ed-7:amd1|file://iec61970cim17v40_iec61968cim13v13a..
versionInfo	String	Public	owl		3.0.0
creator	String	Public	dct		IEC TC57 WG13
description	String	Public	dct		This vocabulary is describing the core equipment profile from IEC 61970-600-2.
identifier	StringIRI	Public	dct		urn:uuid:9fd00587-84d4-4910-bbd4-e003ed7e0821
issued	DateTime	Public	dct		
language	StringFixedLanguage	Public	dct		en-GB
license	String	Public	dct		https://creativecommons.org/licenses/by/4.0/
modified	Date	Public	dct		2020-10-12
publisher	String	Public	dct		IEC
rights	String	Public	dct		Copyright
rightsHolder	String	Public	dct		IEC
title	String	Public	dct		Core Equipment Vocabulary
backwardCompatibleWith	IRI	Public	owl		

Figure 47 – Example of an ontology class in a profile

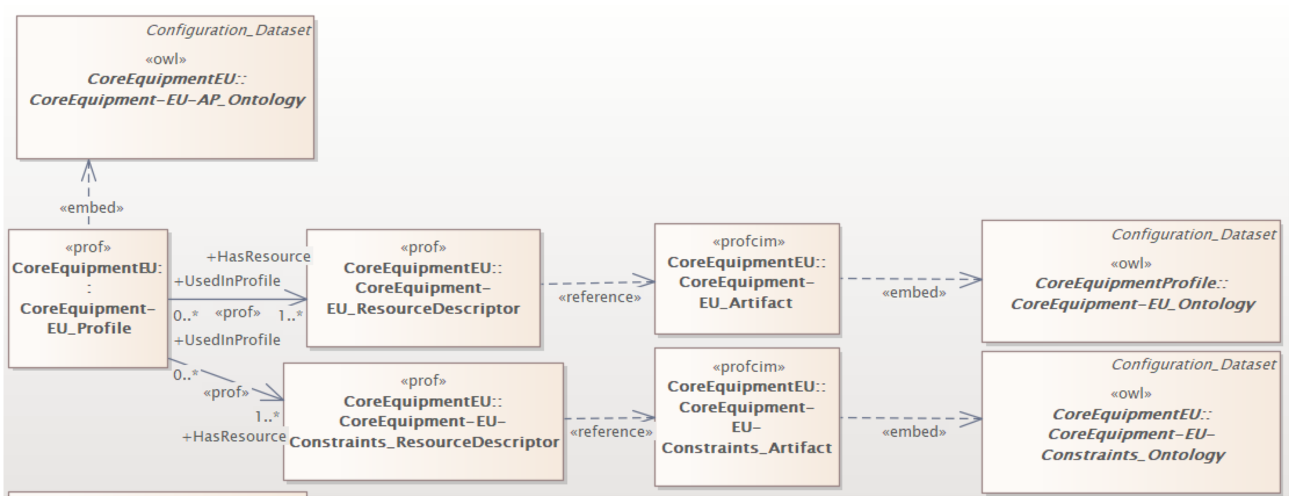


Figure 48 – Example of InfConfiguration for Core Equipment profile in CGMES v3

Maintenance of the profiles using CimConteXtor

Maintenance of the profiles is an important task which shall be done on a regular basis in order to be at the end less time consuming. The canonical CIM (TC57CIM) is available from IEC CIM Model Managers. It is recommended that on regular basis the extensions and profile packages are synchronized with the latest release of canonical CIM.

This section focuses on how available utilities from CimConteXtor are used in combination with regular features from Enterprise Architect in order to maintain the profiles and align them with new versions of canonical CIM. The procedure is used for CGMES profile, but it can also be used for other profiles.

8.1 Procedure to move extensions and profiles to another EA file

The procedure to move extensions and profile packages from one EA file to another includes the following steps:

- Export packages from the source EA file
- Import packages in the target EA file.

These steps are described in detail in the following subsections.

8.1.1 Export all packages from the source EA file

- The reference data shall be exported. The export is launched as shown in Figure 49.

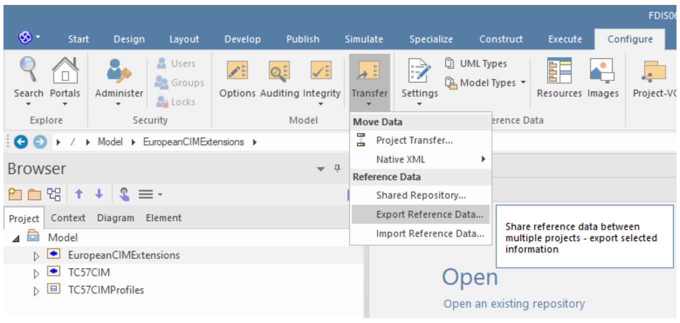
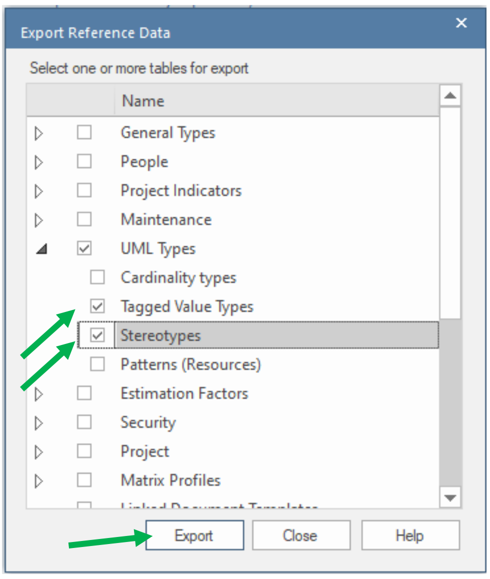


Figure 49 – Launch export of reference data

- The dialog shown in Figure 50 opens. Please select “Tagged Values Types” and “Stereotypes” and click button “Export”. You will be prompted to save the file. Please save the file in a dedicated folder.


Export Reference Data

Select one or more tables for export

Name
☐ General Types
☐ People
☐ Project Indicators
☐ Maintenance
☑ UML Types
☐ Cardinality types
☑ Tagged Value Types
☑ Stereotypes
☐ Patterns (Resources)
☐ Estimation Factors
☐ Security
☐ Project
☐ Matrix Profiles

Figure 50 – Export of reference data

- The package containing the extensions of the canonical CIM shall be exported. XMI export by Enterprise Architect is used. Please note that only the version specified in the figures below is tested. Other versions may have issues in exporting all that is needed.

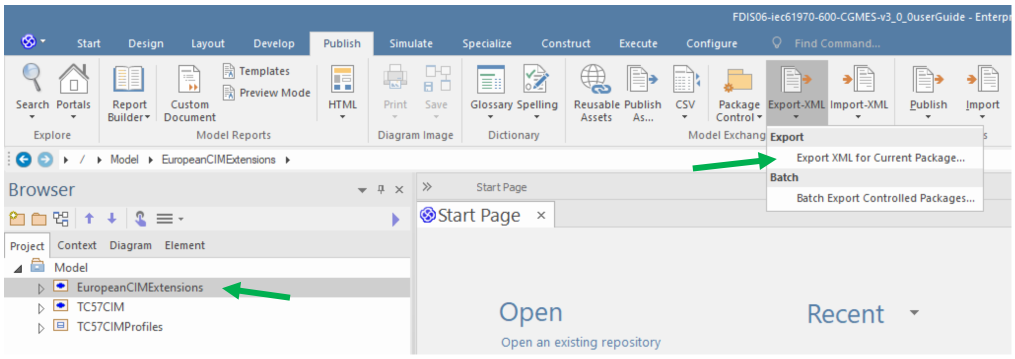
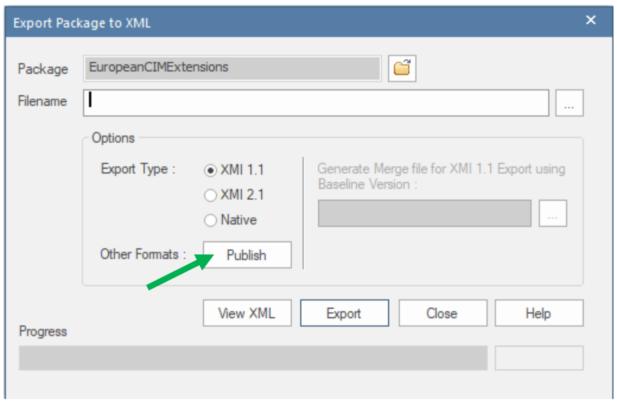


Figure 51 – Launch export of a package to XMI

Figure 50 shows how the export of a package to XMI is launched. Note that the package first needs to be selected and then “Export XML for Current Package…” is selected.
The dialog shown in Figure 52 will open. In this dialog the button Publish shall be selected.


Export Package to XML

Package: EuropeanCIMExtensions

Filename:

Options

Export Type: [✓ XMI 1.1] XMI 2.1 Native

Generate Merge file for XMI 1.1 Export using Baseline Version :

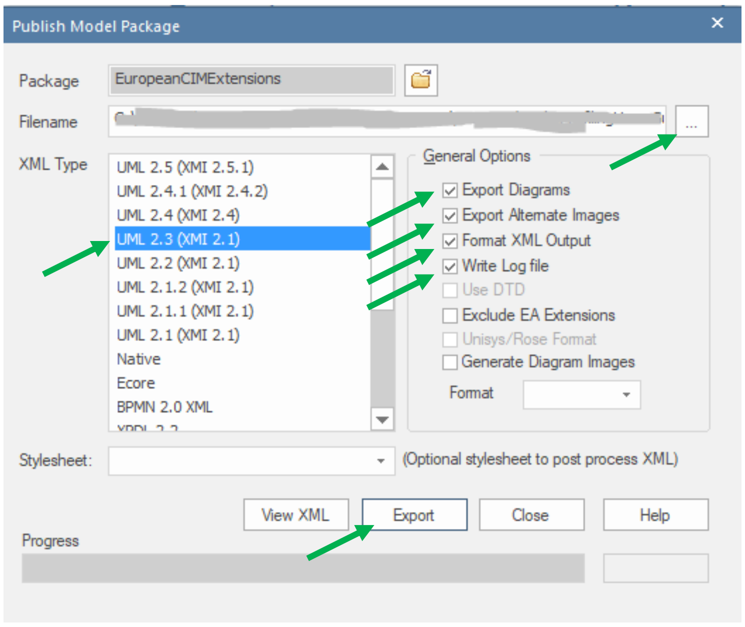
Other Formats : Publish

View XML · Export · Close · Help

Progress

Figure 52 – “Export Package to XML” dialog

The dialog shown in Figure 53 will open. In this dialog the file name and location for the export shall be given by selection the button “…”. It is very important to use XML type “UML 2.3 (XMI 2.1)” and the other selected check boxes. The “Export” button is clicked.


Publish Model Package

Package: EuropeanCIMExtensions

Filename: [redacted]

XML Type: UML 2.5 (XMI 2.5.1), UML 2.4.1 (XMI 2.4.2), UML 2.4 (XMI 2.4), [✓ UML 2.3 (XMI 2.1)], UML 2.2 (XMI 2.1), UML 2.1.2 (XMI 2.1), UML 2.1.1 (XMI 2.1), UML 2.1 (XMI 2.1), Native, Ecore, BPMN 2.0 XML

General Options

- ☑ Export Diagrams
- ☑ Export Alternate Images
- ☑ Format XML Output
- ☑ Write Log file
- ☐ Use DTD
- ☐ Exclude EA Extensions
- ☐ Unisys/Rose Format
- ☐ Generate Diagram Images

Format:

Stylesheet: (Optional stylesheet to post process XML)

View XML · Export · Close · Help

Progress

Figure 53 – “Publish Model Package” dialog

- The package containing the profiles shall be exported. The procedure is exactly the same as for the export of the package containing the extensions or a package containing classes included in IEC CIM standards. The only difference is that another package is selected to be exported.

8.1.2 Import necessary packages to the target EA file

- First the EA file where the exported packages shall be imported needs to be prepared. For example, if the UML from TC57 is iec61970cim17v38_iec61968cim13v12_iec62325cim03v17a.eap the file is copied and renamed with the name of the target EA file, e.g. FDIS06-iec61970-600-CGMES-v3_0_0.eap. Then this file is opened in Enterprise Architect. The file contains only TC57CIM package as shown in Figure 54.

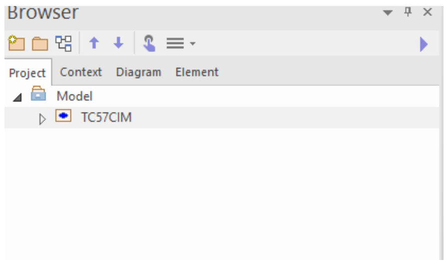


Figure 54 – EA file before import

- The options of the EA file shall be checked as shown in Figure 5. GML and BPMN shall be disabled.
- The reference data is imported as shown in Figure 55. Then the dialog shown in Figure 8 is followed to import the reference data.

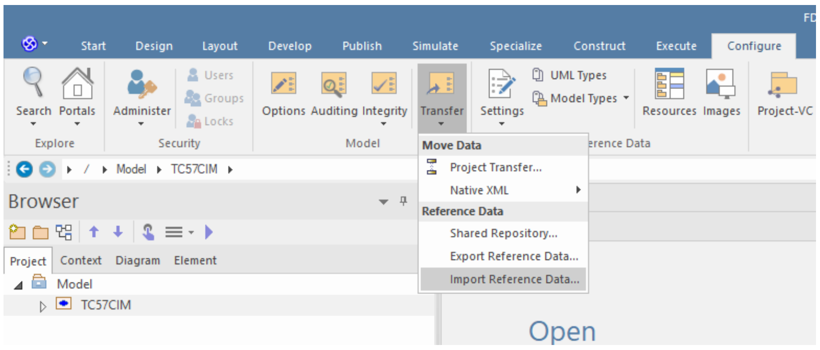


Figure 55 – Launch import of reference data

- The next step is to import the package with the extensions. Note that in case there are no extensions related to the designed profiles this step is skipped. It is important to first import the extensions and not the profiles package because the profiles are dependent from canonical CIM, hence the canonical CIM shall be complete at the time the XMI containing the profiles package is imported.

  The launch of the import is shown in Figure 56.

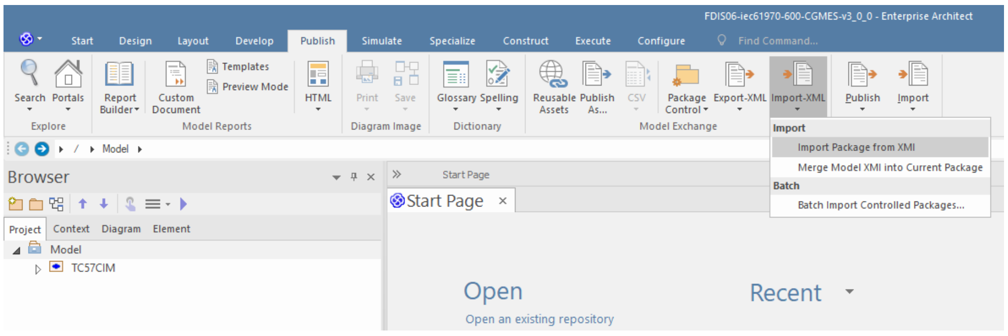


Figure 56 – Launch import of a package

After launching of the import, the dialog “Import Package from XML” opens (Figure 57). The button “…” is used in order to select the xml package. A special attention shall be paid to the checkbox “Strip GUIDs”. If this checkbox is selected all GUIDs will be changed in the imported package. One of the negative effects of this action is that an automatic package or model comparison cannot be performed, i.e. the result will be 100% difference. Therefore, it is recommended that GUIDs are kept persistent. However, there might be some complex profiling activities on big profiles where that option can be used to facilitate the profiling work. Once the profile is completed the GUIDs shall be kept persistent.

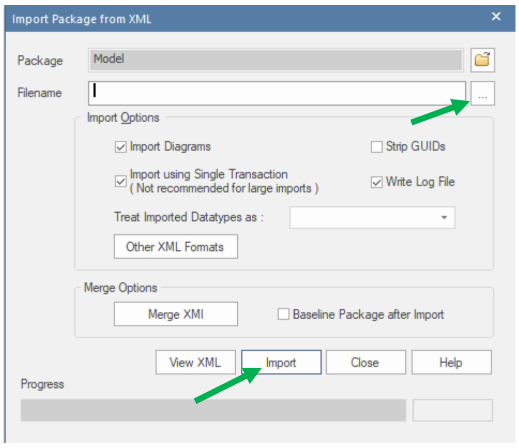
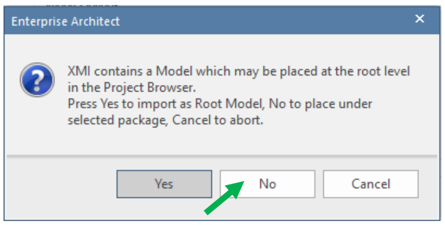


Figure 57 – “Import Package from XML” dialog

When the “Import” button is clicked the dialog shown in Figure 58 opens. It is important to click “No” button as it is required that in the EA file there is only one root model. Once the import is complete button “Close” (Figure 57) is selected.



XMI contains a Model which may be placed at the root level in the Project Browser.
Press Yes to import as Root Model, No to place under selected package, Cancel to abort.

Figure 58 – Import dialog

- The next step is to import the package with the profiles. The procedure is exactly the same as for the import of the package containing the extensions. The only difference is that another package is selected to be imported.

- After the import is completed the order of the packaged shall be adjusted using the buttons as shown in Figure 59. This is necessary in order to achieve the order when the extension package is the first package, followed by TC57CIM and at the end the package that contains the profiles.

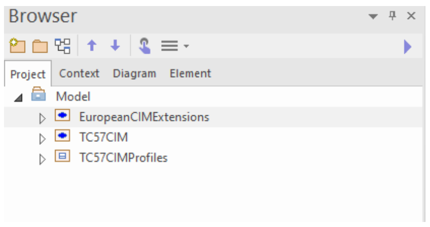


Figure 59 – Ordering of packages

- The last step is to make sure that the package dependency is correct. In order to do this, open the package dependency diagram and verify if all the dependencies are correct. Please note that you should most probably need to recreate the dependency of the profile package with the extension package.

8.2 Understanding the changes between two versions

In order to perform the maintenance of the profiles and in general before starting using an EA file it is important to get familiar with the content of the EA file and the character of changes that were introduced in the newer version. There are different techniques how this can be performed and the best is to utilise multiple of them in order to have a complete picture. Please note that a single technique alone may not give all necessary details.

The techniques are as follows:

- Getting familiar with the change log provided by CIM Model Manager. Normally the CIM Model Manager completes a detail change log what has been changed in the released version. It also includes an updated issue list which indicates the reason of the changes. In case the person that is doing the profiling work has access to IEC modelling weekly calls and meetings that person would already have a good understanding on the character of the changes and the reasons, which helps a lot.

- Compare TC57CIM package in Enterprise Architect. This can be done by either by

    - creating a baseline and then compare with a baseline, or

    - export an XMI and then compare a package with this XMI.

  The following subsections explain the procedure for compare alternatives. In addition, an open source tool CimPal can be used for comparison between two RDFS which will also give information on the differences.

8.2.1 Compare packages by using baselines

The following steps are followed in order to compare using baselines. The example below is for the TC57CIM package, but can be applied for any other package in the EA file.

- Open the EA file where you want to save the baseline.

- Select the package that you want to be a baseline

- Launch the “Manage Baseline…” as shown in Figure 60.

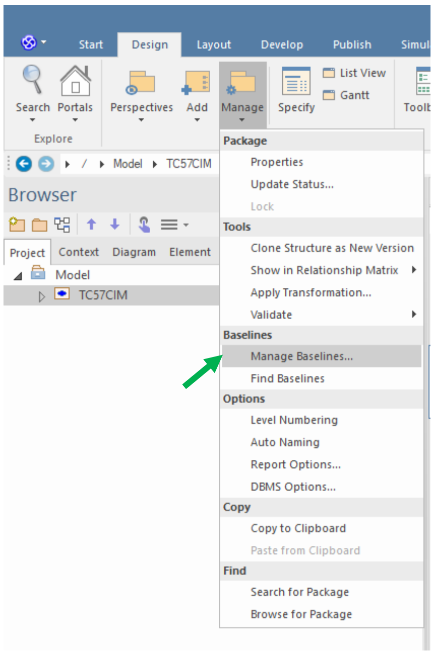
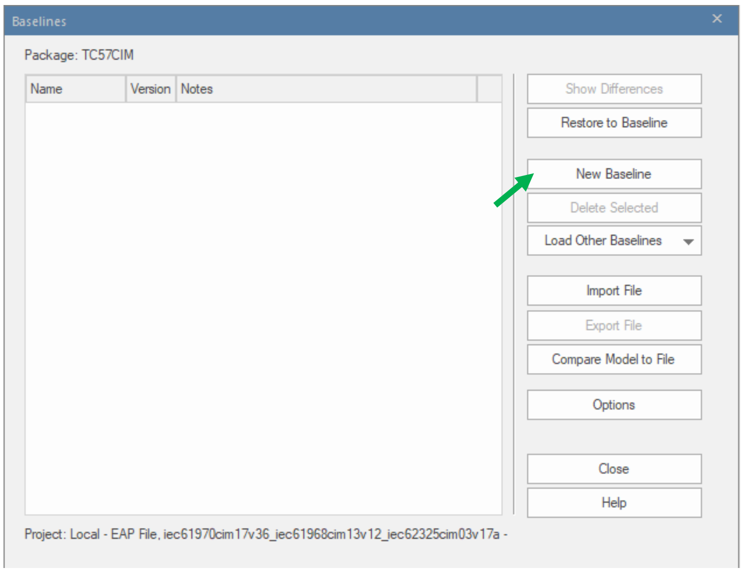


Figure 60 – Launch “Manage Baselines…”

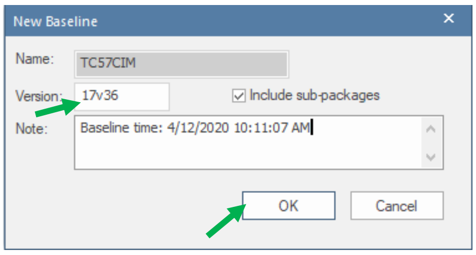
- The dialog as shown in Figure 61 opens. Click button “New Baseline” in order to create the baseline.



Project: Local - EAP File, iec61970cim17v36_iec61968cim13v12_iec62325cim03v17a -

Figure 61 – “Baselines” dialog

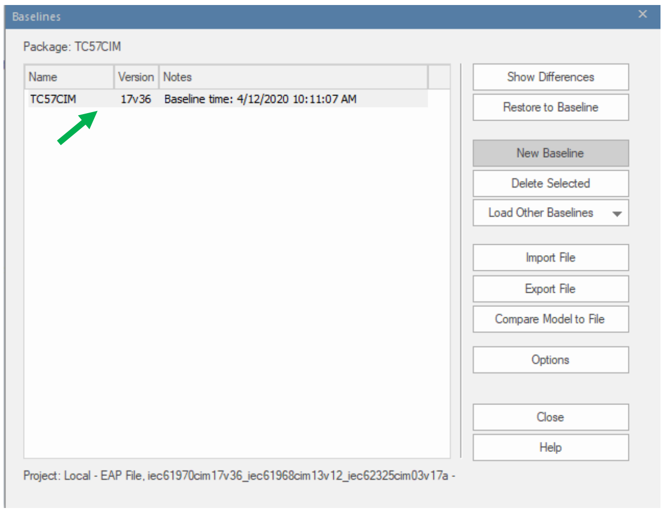
- The dialog “New Baseline” opens as shown in Figure 62. In this dialog the version shall be specified and “OK” button shall be selected. This will launch the creation of the new baseline.



Name: TC57CIM
Version: 17v36
☑ Include sub-packages
Note: Baseline time: 4/12/2020 10:11:07 AM

Figure 62 – “New Baseline” dialog

- When the baseline is created it is seen in the “Baselines” dialog as shown in Figure 63.



Name	Version	Notes
TC57CIM	17v36	Baseline time: 4/12/2020 10:11:07 AM

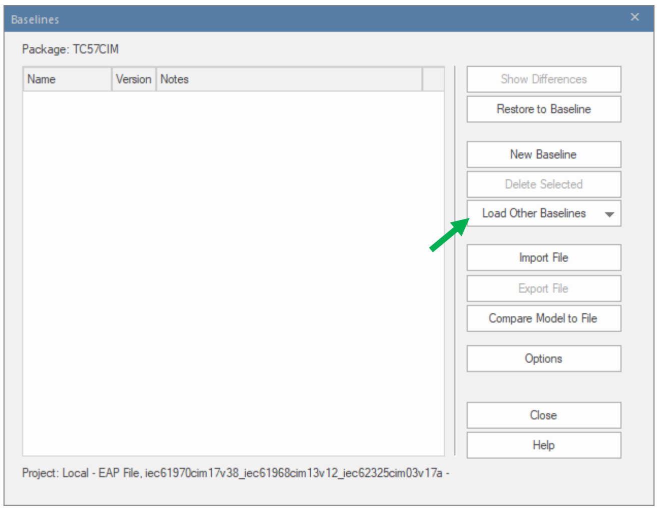
Project: Local - EAP File, iec61970cim17v36_iec61968cim13v12_iec62325cim03v17a -

Figure 63 – List baselines

- The EA file with the baseline can be closed.

- The EA file where the comparison shall be done is opened.

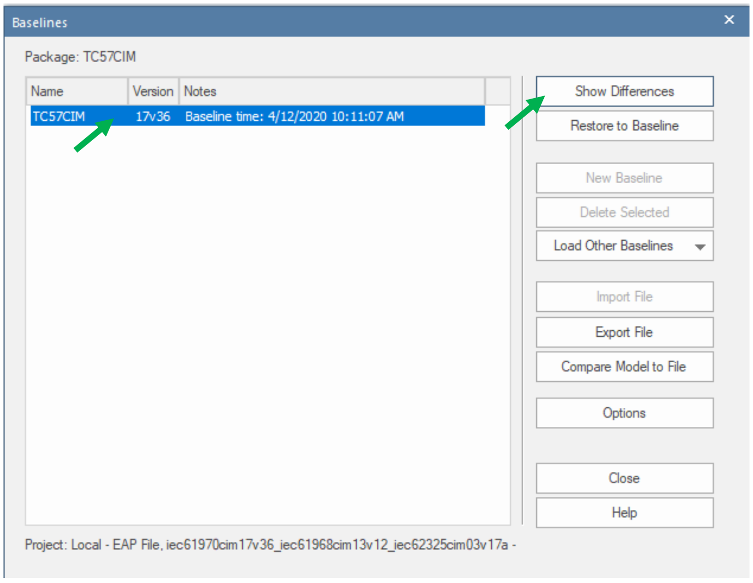
- Select the package that shall be compared. In the example here this is TC57CIM package.
- Launch "Manage Baselines.." as shown in Figure 60. The dialog "Baselines" opens as shown in Figure 64. Select the button "Load Other Baselines" and then select "Load from file". This will open the standard open file dialog where you need to select the EA file where the baseline was saved in the previous steps.



Project: Local - EAP File, iec61970cim17v38_iec61968cim13v12_iec62325cim03v17a -

Figure 64 – "Baselines" dialog for opening of other baselines

- After the selection of the EA file the baseline will be listed as shown in Figure 65. Then select the baseline by clicking on it. This will activate button "Show Differences". Then select the button "Show Differences" to launch the comparison.

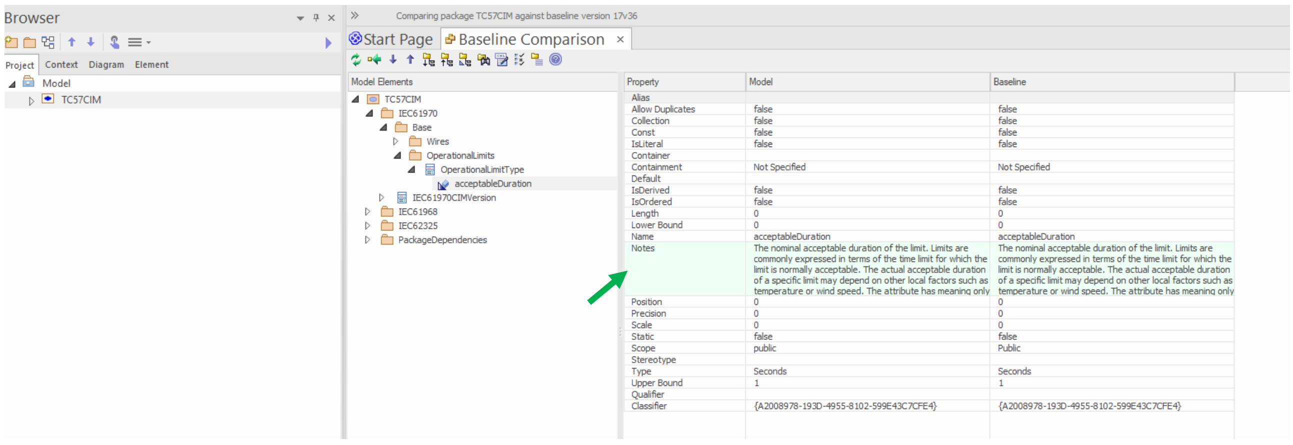


Name	Version	Notes
TC57CIM	17v36	Baseline time: 4/12/2020 10:11:07 AM

Project: Local - EAP File, iec61970cim17v36_iec61968cim13v12_iec62325cim03v17a -

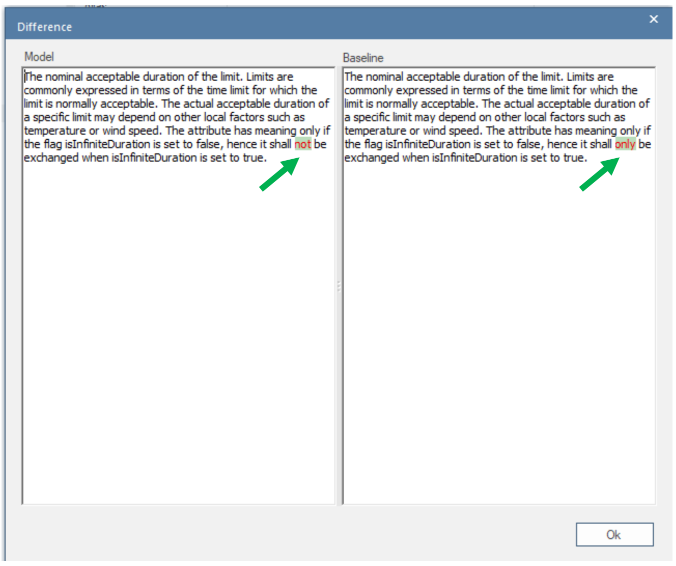
Figure 65 – Compare with a baseline

- After the comparison is finished the report will be shown in Figure 66 the "Baseline Comparison" tab, where the differences can be analysed. The comparison can also be exported from there. Differences are visualised with different colour. By double clicking on the row with the difference, another dialog is shown (Figure 67) where the text difference (in this case) is marked in red. In that specific case this is a difference introduced in CIM17v38 which is the "Model". The baseline is CIM17v36.



Property	Model	Baseline
Alias		
Allow Duplicates	false	false
Collection	false	false
Const	false	false
IsLiteral	false	false
Container		
Containment	Not Specified	Not Specified
Default		
IsDerived	false	false
IsOrdered	false	false
Length	0	0
Lower Bound	0	0
Name	acceptableDuration	acceptableDuration
Notes	The nominal acceptable duration of the limit. Limits are commonly expressed in terms of the time limit for which the limit is normally acceptable. The actual acceptable duration of a specific limit may depend on other local factors such as temperature or wind speed. The attribute has meaning only	The nominal acceptable duration of the limit. Limits are commonly expressed in terms of the time limit for which the limit is normally acceptable. The actual acceptable duration of a specific limit may depend on other local factors such as temperature or wind speed. The attribute has meaning only
Position	0	0
Precision	0	0
Scale	0	0
Static	false	false
Scope	public	Public
Stereotype		
Type	Seconds	Seconds
Upper Bound	1	1
Qualifier		
Classifier	{A2008978-193D-4955-8102-599E43C7CFE4}	{A2008978-193D-4955-8102-599E43C7CFE4}

Figure 66 – Baseline comparison report



Model

The nominal acceptable duration of the limit. Limits are commonly expressed in terms of the time limit for which the limit is normally acceptable. The actual acceptable duration of a specific limit may depend on other local factors such as temperature or wind speed. The attribute has meaning only if the flag isInfiniteDuration is set to false, hence it shall not be exchanged when isInfiniteDuration is set to true.

Baseline

The nominal acceptable duration of the limit. Limits are commonly expressed in terms of the time limit for which the limit is normally acceptable. The actual acceptable duration of a specific limit may depend on other local factors such as temperature or wind speed. The attribute has meaning only if the flag isInfiniteDuration is set to false, hence it shall only be exchanged when isInfiniteDuration is set to true.

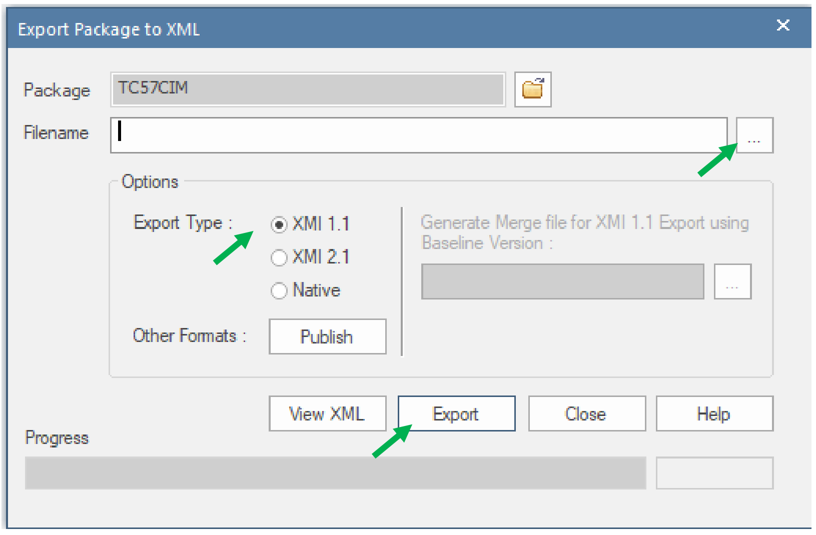
Figure 67 – Showing the difference on an item

8.2.2 Compare packages by using XMI export

The procedure to compare packages by using XMI export basically includes two steps as follows:

- Export of the XMI of the package that shall be compared
- Compare a package with the exported in XMI package

The export of a package in XMI is generally the same as the exports performed in section 8.1.1 however XMI 1.1 is selected for the export and the button “Publish” is not selected as shown in Figure 68.



Export Package to XML

Package: TC57CIM

Filename:

Options

Export Type : [✓ XMI 1.1] XMI 2.1 Native

Other Formats : Publish

Generate Merge file for XMI 1.1 Export using Baseline Version :

Figure 68 – Export XMI for comparison

Once the export is performed the EA file can be closed. Then the EA file where the comparison shall be performed is opened. The comparison is launched as shown in Figure 69.
It is important that the package that shall be compared to the XMI is selected before the selection of the XMI.

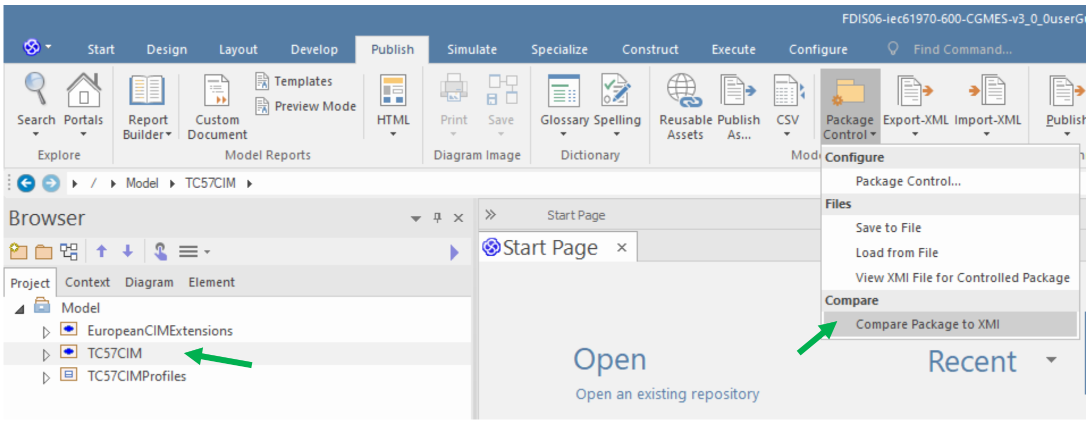


Figure 69 – Launch of XMI based comparison

After the selection of “Compare Package to XMI” you will be asked to select the XMI file (XMI version 1.1) and the comparison will be launched. The result is shown in the same way as in Figure 66.

8.3 Perform integrity check

Integrity check helps to identify potential issues with the profiles. For CGMES the utility “CGMESIntegrityCheck” is used. The utility is only run on a profile package (i.e. per profile not on a group of profiles). It is required that there is a version class as explained in section 7.2.3. That requirement is only valid if using older version of CimConteXtor. The latest version is adapted to also deal with version and ontology classes. The utility is launched as shown in Figure 70.

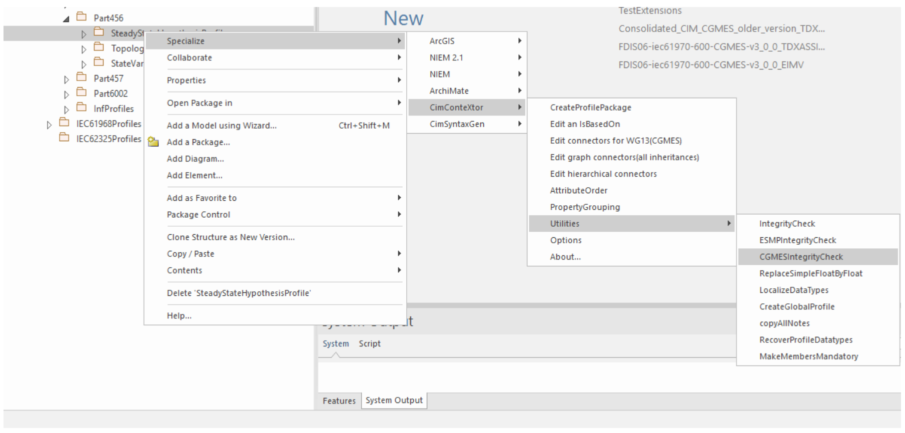


Figure 70 – Launch of “CGMESIntegrityCheck”

The result is in a form of .csv file (CheckError{*name of the profile*}.csv) which is saved in the resources folder of CimConteXtor (..:\Users\...\AppData\Roaming\Zamiren\CimContextor\Ressources).

In order to open the csv file in an easy way the file modelCheckProfile.xlsx is used. This file has a macro that needs to be enabled. The following figures show the procedure to open the csv file.

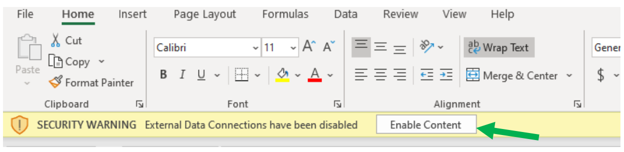
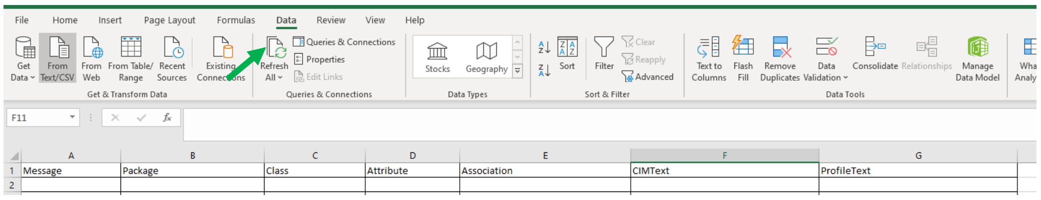

SECURITY WARNING External Data Connections have been disabled
Enable Content

Figure 71 – Enable macro



	A	B	C	D	E	F	G
1	Message	Package	Class	Attribute	Association	CIMText	ProfileText
2							

Figure 72 – Import the csv by using “Refresh All”

There are multiple messages reported by the integrity utility and each message/issue has severity reported. The interpretation of the result can be complicated because actually the utility is also used as a reporting utility so messages reported by the utility can be normal for a profile. The following table summarizes the checks. Please refer to the CimConteXtor manual (Annex I) for the most updated version of the checks.

ID	Integrity check name	Description	Severity
1	**NotIsBasedOnPackage**	It checks if a profile package is IsBasedOn.	Violation
2	**NotUniquePackageName**	It checks if all package names in a profile package are unique.	Violation
3	**NotAGoodProfileOntology**	It checks if profile packages have an Ontology element and only one.	Warning/Violation
4	**NotAGoodProfileVersion**	It checks if profile packages have a version element and only one.	Violation /Warning)
5	**HasOCLconstraint**	It reports the presence of an OCL constraint in an element (package, class, datatype, attribute and association). The content of the constraint is reported. No report for endRoleName	Warning

6	**PackageDescriptionIs**	It reports the description of the profile package. There is no consistency check.	Info
7	**NotGoodPackageNames**	It checks if the names of packages in a profile package have corresponding CIM package names. It also checks if the package is from extension.	Warning
8	**DuplicateElementName**	It checks if in a profile package an element (class, datatype) name is unique.	Violation
9	**HasNonAttachedNote**	It checks if there is a diagram note which is not attached to an element (class or association). The content of the note is reported.	Warning
10	**NotIsBasedOn**	It checks if a profile element (class, attribute, association) is based on a corresponding CIM element given by a GUIDBasedOn.	Violation
11	**NotSameDescription**	It checks if the element (class, datatype, association, attribute) has a different description that the one in CIM. For end role description see "NotSameEndRoleDescription".	Violation
12	**HasMoreThanOneIsBasedOnLink**	It checks if an element (package, class, datatype) has more than one isBasedOn dependency link.	Violation
13	**HasAnUnkownLink**	It checks if a link is of kind notelink, association, aggregation, generalization.	Violation
14	**AttributeInconsistency**	It checks if an element attribute has no corresponding attribute in CIM as it might have been deleted in the CIM.	Violation
15	**NotSameStereotypes**	It checks if a profile element (class, datatype, attribute, enumeratedLiteral – association) has different stereotypes than its CIM counterpart.	Warning
16	**NotProperlyTyped**	It checks if the type of an attribute is referring to an unknown datatype or enumeration.	Violation
17	**NotInProfileDatatype**	It checks if the type of a profile attribute does not refer to a profile datatype (CIMDatatype, Primitive) or a profile enumeration, usually in that case, the type is referring to a CIM Domain datatype or enumeration.	Violation
18	**NotConformedPrimitive**	It checks if the type of the CIMDatatype value attribute that should be a primitive does not have the corresponding stereotype.	Violation
19	**NotSameType**	It checks if the type of an attribute is different from the type of its CIM counterpart.	Violation
20	**NotConsistantFeature**	It checks if the attribute has not the same « isConst » or « isStatic » feature than its CIM counterpart.	Violation
21	**InconsistantEnumeratedLiteral**	It checks if a profile enumeration member has a counterpart in the CIM corresponding enumeration.	Violation
22	**NotConsistantEnumerationInitialValues**	It checks if the initial value of an attribute is not present in the corresponding profile enumeration.	Violation
23	**NotSameInitialValue**	It checks if the initial value of an attribute in a profile is different from the initial value of its CIM counterpart.	Warning
24	**NotSameCardinality**	It checks if the cardinality of a profile attribute is different from its CIM corresponding one.	Warning
25	**NotSameName**	It checks if the name of an element (class, attribute, datatype) is different from its CIM corresponding one (given by the GuidBasedOn TaggedValue).	Violation
26	**MissingGUIDBasedOnTag**	It checks if a profile element (Class, datatype, association, attribute) has no GUIDBasedOn TaggedValue or if this TaggedValue is empty.	Violation

27	**ElementInconsistancy**	It checks if a given GUIDBasedOn TaggedValue of a profile element (Class, datatype, association) corresponds to a CIM element.	Violation
28	**NotIsBasedOnLink**	It checks if a profile element (class, datatype) has no IsBasedOn dependency link with its CIM counterpart.	Violation
29	**NotValidIsBasedOnLink**	It checks if a profile element (class, datatype) has an isBasedOn dependency link to a CIM element that is not its CIM counterpart.	Violation
30	**AssociationEndsInconsistency**	It checks for a profile association IsBasedOn a parent CIM association if the source classes are related (same hierarchy) and if the target classes are related (same hierarchy).	Violation
31	**NotSameEndRoleName**	It checks if profile association endRole names are the same as their CIM counterparts.	Violation
32	**NotSameEndRoleDescription**	It checks if profile association end role's descriptions are the same as their CIM counterparts.	Violation
33	**NotSameMultiplicity**	It checks if the multiplicities of the end role of a profile association and of its CIM counterpart are different.	Warning
34	**NotExpectedDirection**	It checks if a profile association is correctly oriented. If the association is oriented from cardinality (n..*) towards 0..1 or 1 it is normal, If the association is oriented from cardinality 0..1 towards 0..1 it has to be reported If the association is orientated from cardinality 1 towards n..* it has to be reported.	Info
35	**AssociationNameNotEmpty**	It checks if the name of an association is not empty. This rule is applied to association name that in CIM is empty by definition. So, the test is to see if the profile association conforms to this.	Violation
36	**HasEmptyNote**	It checks if profile element (class, attribute, association, package) notes are not empty. The check is not made for CIMDatatype attribute. For association end role, the result of the test is "NotSameEndRoleDescription".	Violation
37	**NotSameInheritance Chain**	It checks if the inheritance path for profile is the same than the one of their corresponding CIM counterpart.	Violation
41	**InfoOnCheck**	Data about checked profile version: profileVersion, CIMBasedOn packages Version.version concatened	Info
42	**hasAttachedNote**	Info that the class has an attached note	Info
43	**NotSameAbstract**	Checks is the class has the same abstraction than its CIM parent	Info
44	**HasInheritanceIssue**	A class has an ancestor in the profile but this ancestor has no CIM's parent Or The anscestor of the CIM's parent class is different from the ancestor of its ancestor's CIMparent.	Violation
45	**NotAccessibleElement**	A found element (Class, association) given by its GUID is no longer present in the repository It may have been deleted in CIM.	Violation
46	**NotProperlyBasedOn**	A CIMDatatype must be Based on a CIMDatatype	Violation

8.4 Recover profile datatypes

The datatypes shall be local in the profile package and all attributes shall use the datatypes from the profiled classes and not from canonical CIM. In order to achieve this the utility "RecoverProfileDatatypes" is used. The utility is launched as shown in Figure 73.

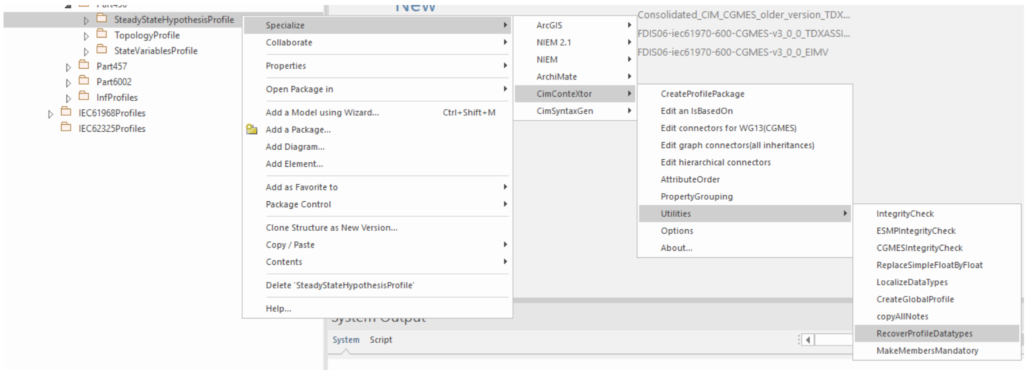


Figure 73 – Launch of utility “RecoverProfileDatatypes”

Please note that before the utility is launched in cases where the profile does not contain all datatypes either they need to be added using CimConteXtor or if there is a general domain profile the utility “LocalizeDataTypes” can be used to add the datatypes to the profile.

8.5 Update descriptions of UML elements

The descriptions of all elements in the profile shall be the same as the descriptions in the canonical CIM (including extensions). In order to achieve this in an easy way the utility “CopyAllNotes” is used. This utility follows the link “IsBasedOn” and updates the descriptions in the profile, so they are the same as in the canonical CIM. The utility is launched as shown in Figure 74. The utility is run on a package which contains a single profile.

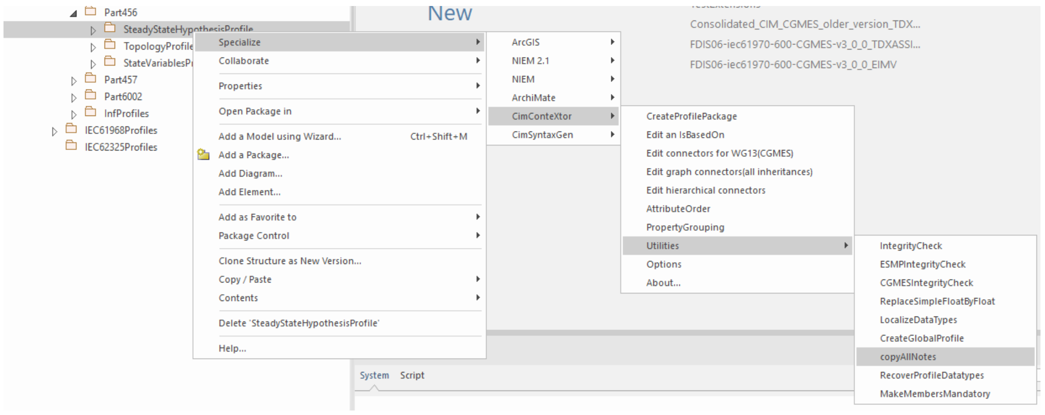


Figure 74 – Launch of utility “CopyAllNotes”

8.6 Compact EA file

The EA file needs to be compacted on regular basis e.g. when some packages are added, deleted, or after some profiling work. This is achieved by using functionality by Enterprise Architect as shown in Figure 75.

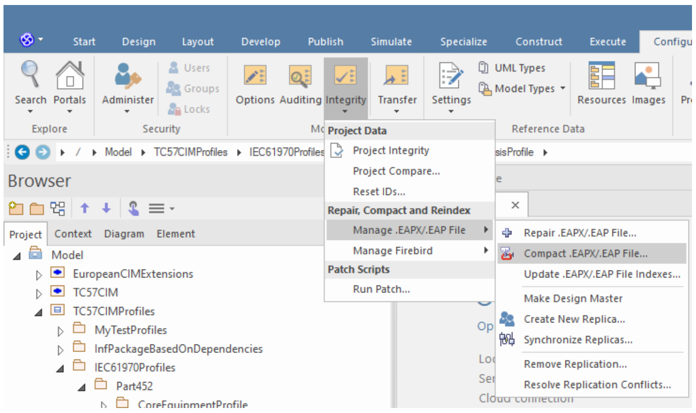


Figure 75 – Launch of “Compact .EAPX/.EAP File...”

Generation of standards and machine-readable documentation

9.1 General overview

The generation of standards and machine-readable documentation is performed by CimSyntaxGen and jCleanCim. jCleanCim shall be run to be sure that the model validates and there are no critical issues found.

The following documentation is normally generated for CGMES:

- The standard for the profile or extensions – as MS Word document. Note that this can be exported by jCleanCim (the preferred way) or by CimSyntaxGen and using a MS Word macro. The formatting/way of looking of the two different exports could be slightly different.
- HTML documentation. Note that there are two different views of this: the HTML exported by CimSyntaxGen and the HTML by Enterprise Architect.
- RDFS and SHACL – which is exported by CimSyntaxGen.

This user guide does not cover the export and validation by jCleanCim as this is already explained in the jCleanCim tutorial.

9.2 Generate HTML using CimSyntaxGen

Before generation of the HTML the configuration of CimSyntaxGen shall be checked either by reviewing the config file (CimSyntaxgen-Config.xml) which is available here: …:\Users\...\AppData\Roaming\Zamiren\CimSyntaxGen\Ressources or by using the GUI. Section 0 in this document provides some examples for configuration file for CGMES 2.4 and CGMES v3.0. The most important part for HTML generation are the values for the stereotypes as if they are not checked the classes, attributes, associations that are marked with the unchecked stereotype will not be exported. The generation of HTML document for a profile is launched as shown in Figure 76.

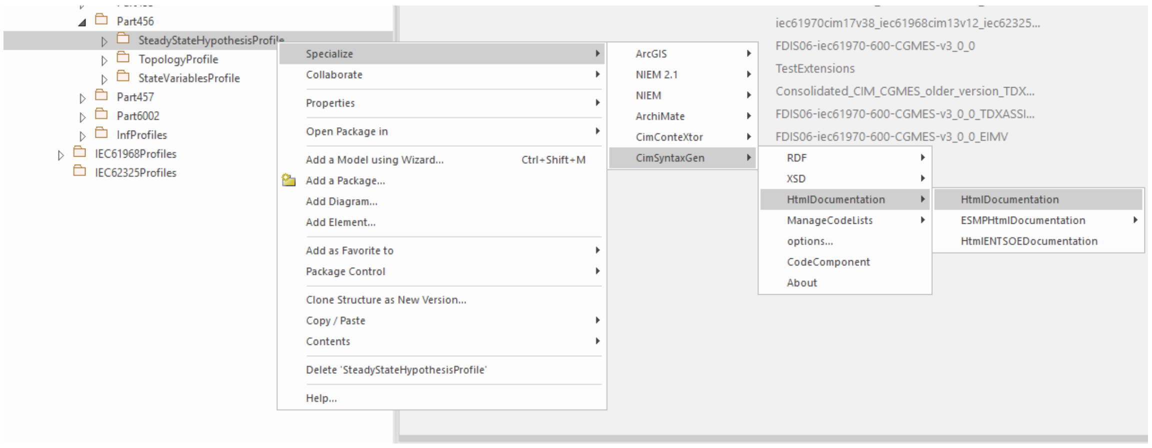


Figure 76 – Launch of HTML export by CimSyntaxGen

When the export is launched you will be asked to give name and location for the export. Then an information as shown in Figure 77 will follow. You need to select "OK" button.

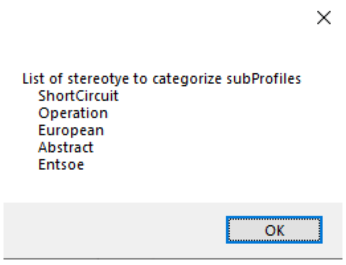

List of stereotye to categorize subProfiles
ShortCircuit
Operation
European
Abstract
Entsoe

Figure 77 – HTML export by CimSyntaxGen – stereotypes information

9.3 Generate MS Word document using CimSyntaxGen

As explained in section 9.2 the options regarding the stereotypes shall be checked and corrected if necessary.

Please note that the procedure below could be different for different versions of MS Word. In addition, the macro could have some problems depending on regional settings and version of the MS Word.

In order to generate the MS Word document using CimSyntaxGen the following steps are performed:

- Launch the export of an HTML document as shown in Figure 78. Please note that this export is different compared to what is exported when following the procedure explained in section 9.2.

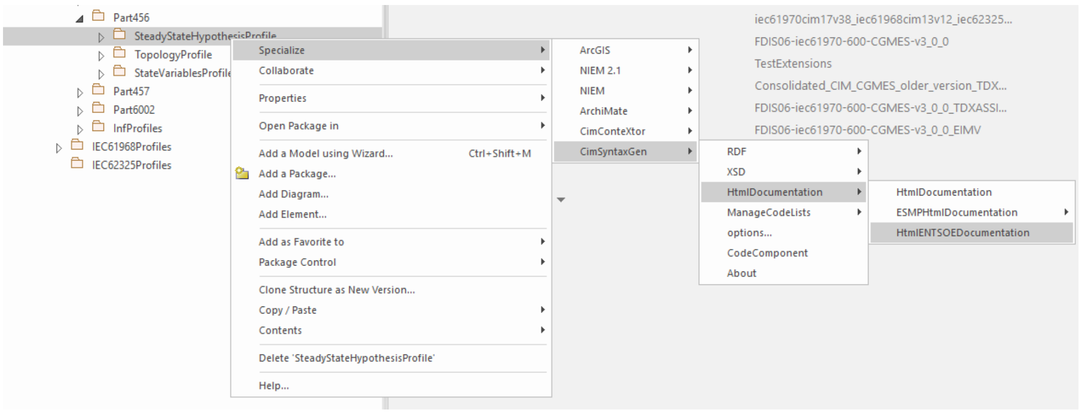
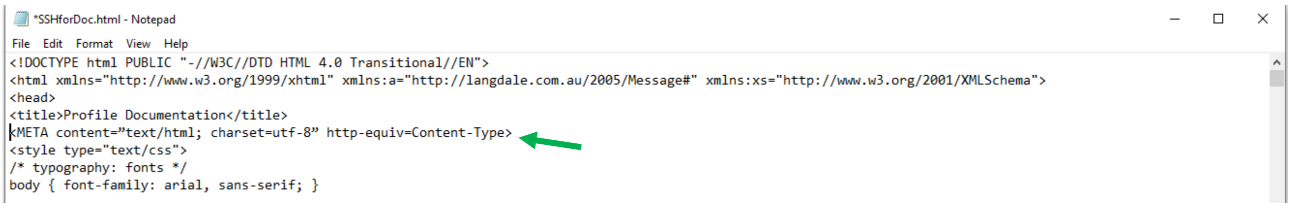


Figure 78 – Launch of HTML export for documentation

- When the export is launched you will be asked to give name and location for the export.

- Open the exported html in Notepad and add the following line "<META content="text/html; charset=utf-8" http-equiv=Content-Type>" after <title> line as shown in Figure 79. Save the document in Notepad (still as html). If this is not done the file will not be treated as UTF-8 encoding and the end result will have bad characters. This is CimSyntaxGen issue which is expected to be fixed in next releases.



```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:a="http://langdale.com.au/2005/Message#" xmlns:xs="http://www.w3.org/2001/XMLSchema">
<head>
<title>Profile Documentation</title>
<META content=”text/html; charset=utf-8” http-equiv=Content-Type>
<style type="text/css">
/* typography: fonts */
body { font-family: arial, sans-serif; }
```

Figure 79 – Fix of the encoding

- The exported html file is then opened with MS Word as shown in Figure 80. After selecting "Choose another app" you need to select (or find and select) Word.

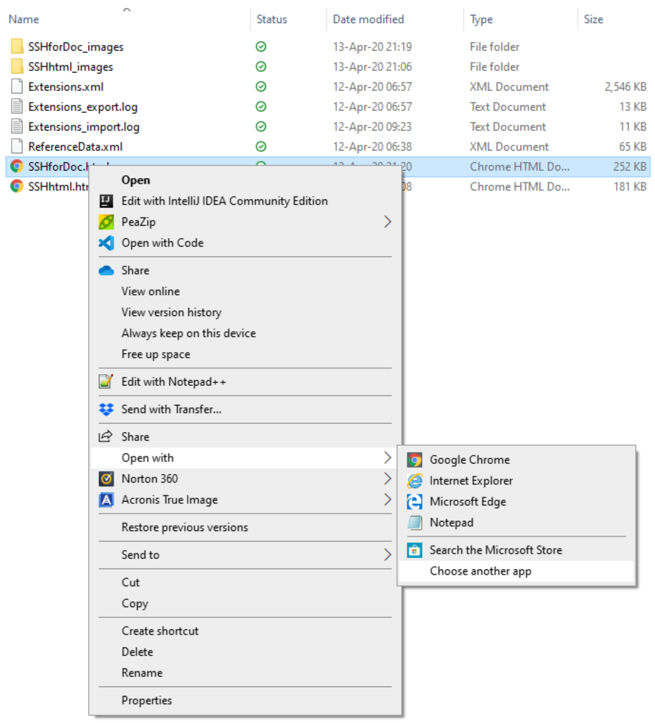


Figure 80 – Open the html with MS Word

- When the file opens in MS Word you need to do Save as .docx as shown in Figure 81.

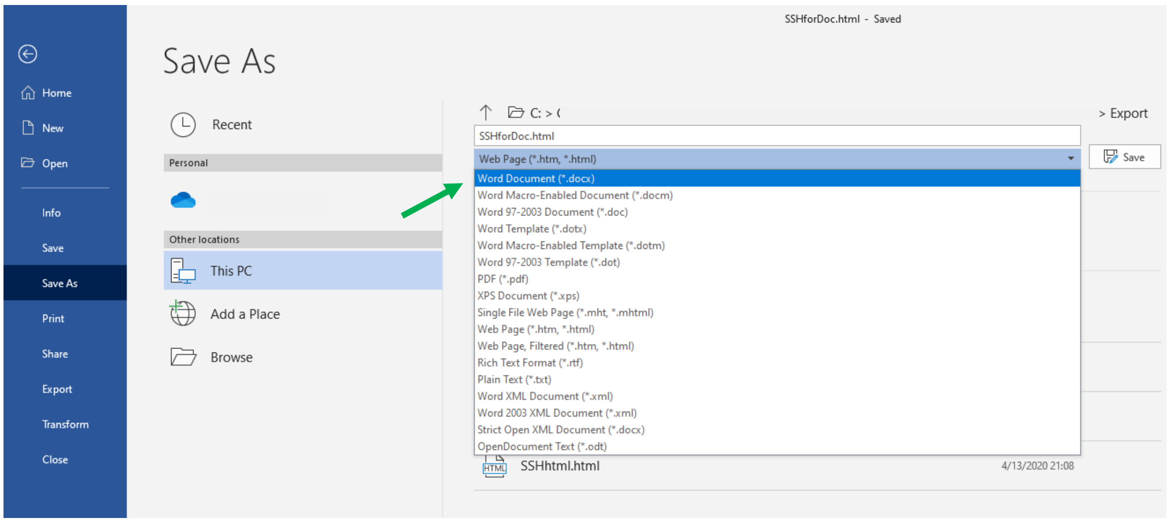
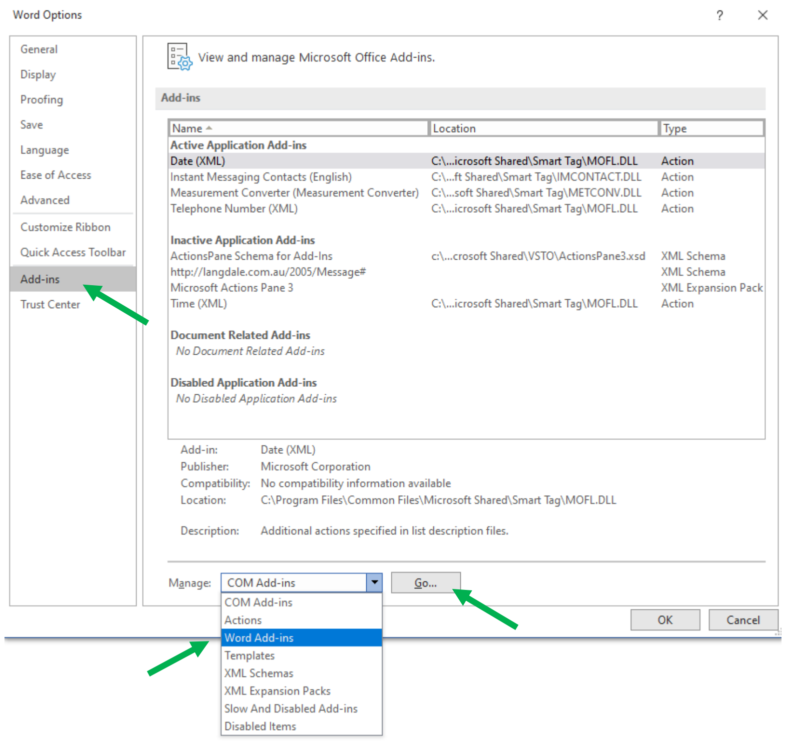


Figure 81 – MS Word “Save as”

- Close the file, the MS Word.
- Open the .docx file that was saved.
- Go to menu File, then Options and then select Add-ins -> Word Add-ins as shown in Figure 82. Select button “Go…”.



View and manage Microsoft Office Add-ins.

Add-ins

Name	Location	Type
Active Application Add-ins		
Date (XML)	C:\...icrosoft Shared\Smart Tag\MOFL.DLL	Action
Instant Messaging Contacts (English)	C:\...ft Shared\Smart Tag\IMCONTACT.DLL	Action
Measurement Converter (Measurement Converter)	C:\...soft Shared\Smart Tag\METCONV.DLL	Action
Telephone Number (XML)	C:\...icrosoft Shared\Smart Tag\MOFL.DLL	Action
Inactive Application Add-ins		
ActionsPane Schema for Add-Ins	c:\...crosoft Shared\VSTO\ActionsPane3.xsd	XML Schema
http://langdale.com.au/2005/Message#		XML Schema
Microsoft Actions Pane 3		XML Expansion Pack
Time (XML)	C:\...icrosoft Shared\Smart Tag\MOFL.DLL	Action
Document Related Add-ins		
No Document Related Add-ins		
Disabled Application Add-ins		
No Disabled Application Add-ins		

Add-in:	Date (XML)
Publisher:	Microsoft Corporation
Compatibility:	No compatibility information available
Location:	C:\Program Files\Common Files\Microsoft Shared\Smart Tag\MOFL.DLL
Description:	Additional actions specified in list description files.

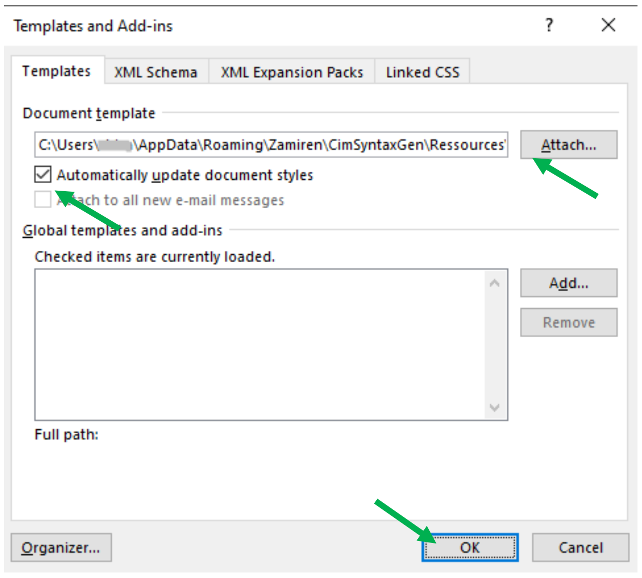
Manage: COM Add-ins

Figure 82 – Manage Add-ins in MS Word

- The dialog shown in Figure 83 will open. The checkbox “Automatically update document styles” shall be checked. Select button “Attach…” and add the file “iec_entsoe_V16.dotm” which is in CimSyntaxGen resources folder here:

  …:\Users\...\AppData\Roaming\Zamiren\CimSyntaxGen\Ressources

  At the end click “OK”.



Document template

C:\Users\ \AppData\Roaming\Zamiren\CimSyntaxGen\Ressources

☑ Automatically update document styles

☐ Attach to all new e-mail messages

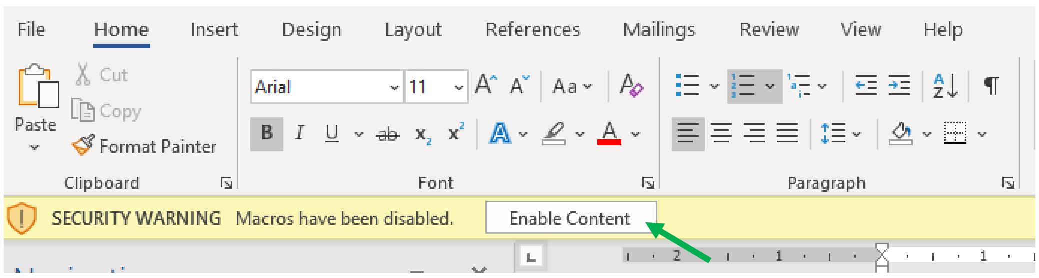
Global templates and add-ins

Checked items are currently loaded.

Full path:

Figure 83 – “Templates and Add-ins” dialog

- Enable the macro by selecting “Enable Content” as shown in Figure 84.



SECURITY WARNING Macros have been disabled.

Figure 84 – Enable macro in MS Word

- The view the macro as shown in Figure 85. Then the dialog show in Figure 86 will open. Select “apourENTSOE” and then select “Run” button. Then wait until finishes, select “End” when the GUI appears. At the end close MS Word and you can open again the file which is ready.

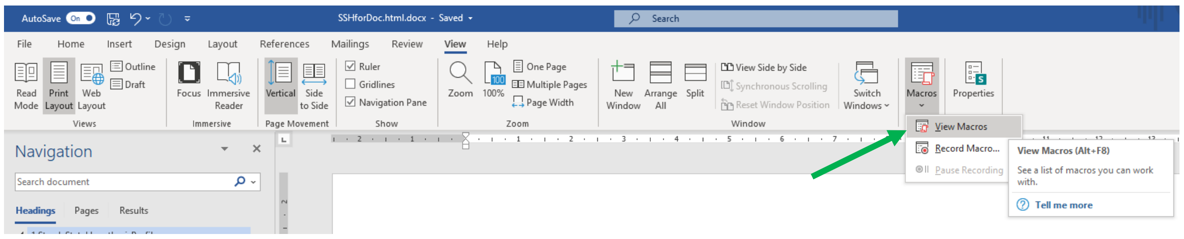
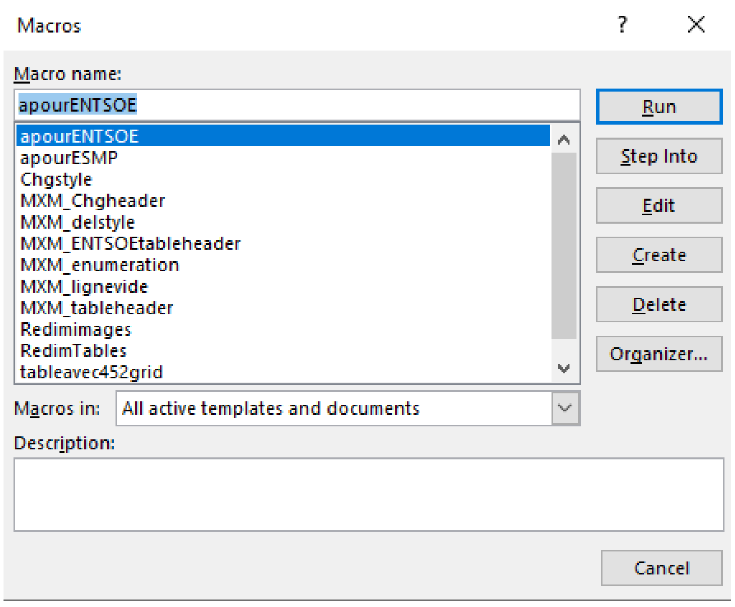


Figure 85 – Launch view macro in MS Word



Macro name: apourENTSOE

apourENTSOE
apourESMP
Chgstyle
MXM_Chgheader
MXM_delstyle
MXM_ENTSOEtableheader
MXM_enumeration
MXM_lignevide
MXM_tableheader
Redimimages
RedimTables
tableavec452grid

Macros in: All active templates and documents

Description:

Figure 86 – “Macros” dialog in MS Word

9.4 Generate RDFS using CimSyntaxGen

Before generation of the RDFS the configuration of CimSyntaxGen shall be checked either by reviewing the config file (CimSyntaxgen-Config.xml) which is available here: …:\Users\...\AppData\Roaming\Zamiren\CimSyntaxGen\Ressources or by using the GUI. Section 0 in this document provides some examples for configuration file for CGMES 2.4 and CGMES v3.0. It is preferred to use the config file in order to setup the export. The following parts of the config file shall be checked and corrected if needed:

- The part related to the URI and prefix of the namespace

<profdata name="DefaultRootModelURI" value="http://iec.ch/TC57/CIM100#" />

<profdata name="RootModelURI" value="http://iec.ch/TC57/CIM100#" />

<profdata name="Prefix" value="cim" />

- The part related to the stereotypes present in the model. Note that properties “ListStereoNamspaceExportable” and “EntsoeEquipmentProfile” are only applicable when exporting CGMES 2.4. In CGMES v3.0 they do not play a role as the Equipment profile is split in an explicit way and not by using ShortCircuit and Operation stereotypes.

<profdata name="ListStereoNamespace" value="European|Abstract|Description" />

<profdata name="ListStereoNamspaceExportable" value="ShortCircuit|Operation" />

<profdata name="EntsoeEquipmentProfile" value="EquipmentCompleteProfile" />

- When the setup of the profiles uses a common domain profile for the datatypes the name of the domain profile is referred in the following property. Note that in CGMES v3.0 profiles all datatypes are explicitly profiles per package.

<profdata name="EntsoeDataTypesDomain" value="DomainProfile" />

- The part were the stereotype for the extension, the version and the name of the extension package is defined.

<profdata name="EntsoeExpStereo" value="European" />

<profdata name="PackageExtension" value="EuropeanCIMExtensions" />

<profdata name="EntsoeVersion" value="ExtIEC61970Version" />

- The part where the prefixes and URI for different stereotypes is defined. Note that stereotype “Abstract” is only used in Dynamics profile and it is deleted in CGMES v3. Stereotype “Description” is used for distinguishing rdf:id and rdf:about. However, in future this may not be used as well.

<profstereo name="Abstract" prefix="cim" uri="http://iec.ch/TC57/CIM100#" html="Checked" rdfs="Checked" />

<profstereo name="Description" prefix="cim" uri="http://iec.ch/TC57/CIM100#" html="Checked" rdfs="Checked" />

<profstereo name="European" prefix="eu" uri="http://iec.ch/TC57/CIM100-EuropeanExtension/1/0#" html="Checked" rdfs="Checked" />

The launch of the RDFS export is performed as shown in Figure 87. There are four different RDFS exports that are supported:

- IEC 61970-501:2006 – this is the Ed1 of the standard, but it is practically not used for CGMES
- IEC 61970-501:2006 augmented (2019) – this is the former “RDFS (augmented)” export that was used for CGMES v2.4 and later on. Bug were fixed in that export. In addition, a copyright notice is added in the exported RDFS.
- IEC 61960-501:2006 augmented (2020). The difference between this export and “IEC 61970-501:2006 augmented (2019)” is that in “IEC 61960-501:2006 augmented (2020)” the ontology class is exported as a header of the RDFS export and not as a regular abstract class. This export is only used when the profile has an ontology class.
- IEC 61970-501:Ed2 – This is the draft version of the Ed2 of the standard. It contains many modifications which are described in the CD of the IEC 61970-501:Ed2. In addition, it contains

the export of SHACL shapes that are derived from the UML (the profile), i.e. SHACL shapes/constraints representing checks for cardinalities and datatypes.

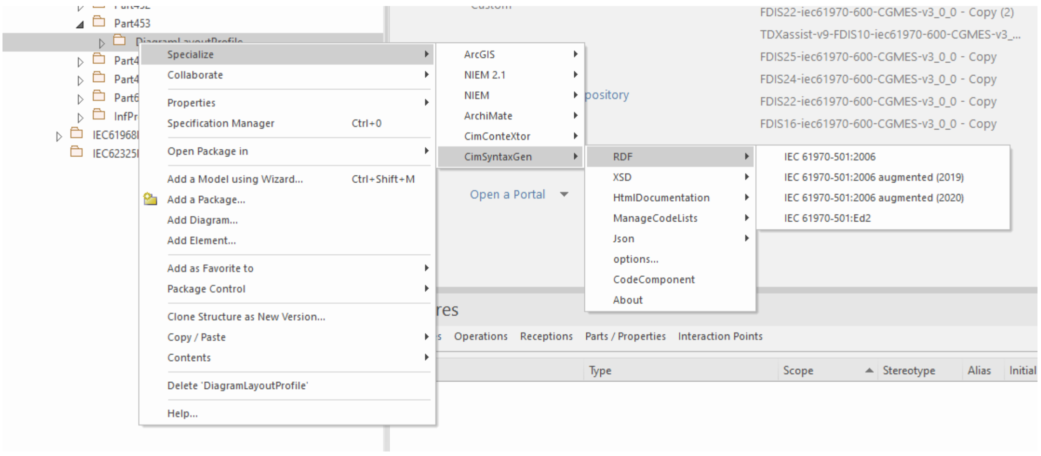
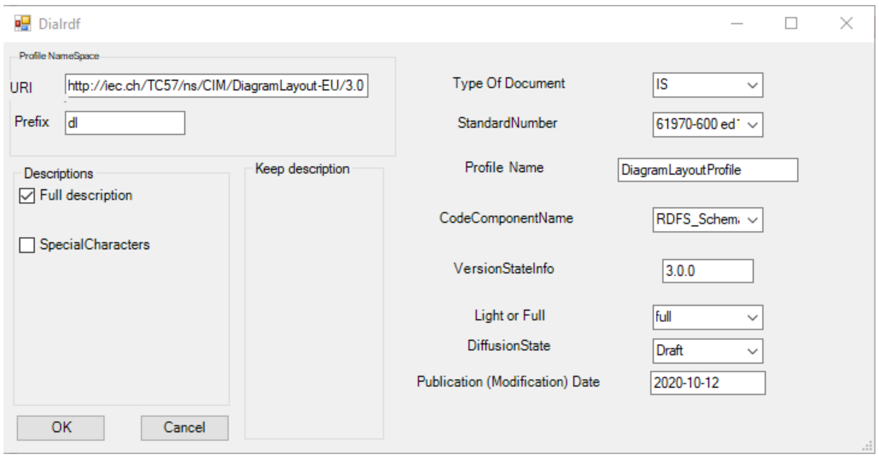

Part453
Specialize
Collaborate
Properties
Specification Manager Ctrl+0
Open Package in
Add a Model using Wizard... Ctrl+Shift+M
Add a Package...
Add Diagram...
Add Element...
Add as Favorite to
Package Control
Clone Structure as New Version...
Copy / Paste
Contents
Delete 'DiagramLayoutProfile'
Help...
ArcGIS
NIEM 2.1
NIEM
ArchiMate
CimContextor
CimSyntaxGen
RDF
XSD
HtmlDocumentation
ManageCodeLists
Json
options...
CodeComponent
About
IEC 61970-501:2006
IEC 61970-501:2006 augmented (2019)
IEC 61970-501:2006 augmented (2020)
IEC 61970-501:Ed2
FDIS22-iec61970-600-CGMES-v3_0_0 - Copy (2)
TDXassist-v9-FDIS10-iec61970-600-CGMES-v3_...
FDIS25-iec61970-600-CGMES-v3_0_0 - Copy
FDIS24-iec61970-600-CGMES-v3_0_0 - Copy
FDIS22-iec61970-600-CGMES-v3_0_0 - Copy
FDIS16-iec61970-600-CGMES-v3_0_0 - Copy
Open a Portal
Operations Receptions Parts / Properties Interaction Points
Type Scope Stereotype Alias Initial

Figure 87 – Launch of RDFS export

The dialog as shown in Figure 89 will open. This dialog gives options to select if all descriptions need to be exports (if Full description is selected) or not as well as some other information which is used for the copyright notice. For details, please check the user guide. The dialog containing the stereotypes and their prefixes and URI will open as shown in Figure 89. It is important to uncheck the stereotypes that relate to the extensions. If this is not done, they will be assigned to the TC57CIM, i.e. will have “cim” prefix and the related stereotype. When the checkbox for the extension is unchecked, the information completed in the config file will be filled in. Then “OK” button is selected. At the end you will be asked to save the file.


Dialrdf

Profile NameSpace
URI: http://iec.ch/TC57/ns/CIM/DiagramLayout-EU/3.0
Prefix: dl

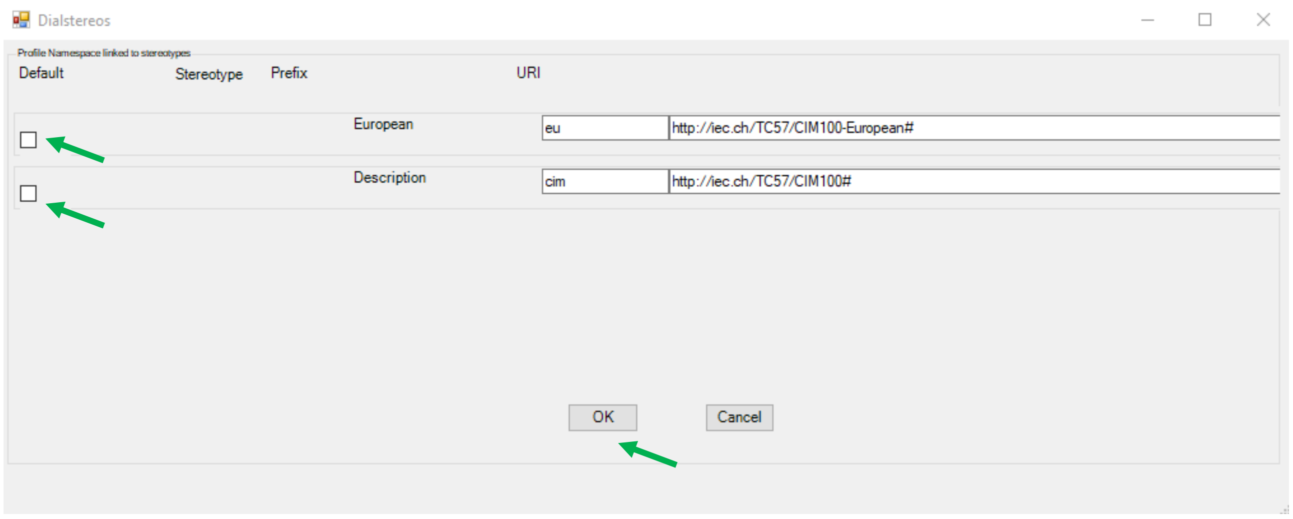
Descriptions
☑ Full description
☐ SpecialCharacters

Keep description

Type Of Document: IS
StandardNumber: 61970-600 ed
Profile Name: DiagramLayoutProfile
CodeComponentName: RDFS_Schem
VersionStateInfo: 3.0.0
Light or Full: full
DiffusionState: Draft
Publication (Modification) Date: 2020-10-12

OK Cancel

Figure 88 – Confirm the namespace of the profile and copyright information


Dialstereos

Profile Namespace linked to stereotypes

Default	Stereotype	Prefix	URI
☐	European	eu	http://iec.ch/TC57/CIM100-European#
☐	Description	cim	http://iec.ch/TC57/CIM100#

Figure 89 – Confirm the stereotypes and their namespace information

9.5 Generate HTML using Enterprise Architect

The advantage of having the HTML generated by Enterprise Architect is that the exported HTML looks very similar to the way of browsing the model in Enterprise Architect. Therefore, it is an alternative to check the model without having Enterprise Architect installed.

The package that shall be exported needs to be selected and the export is launched as shown in Figure 90. Please note that in this way you can export the whole package containing all profiles.

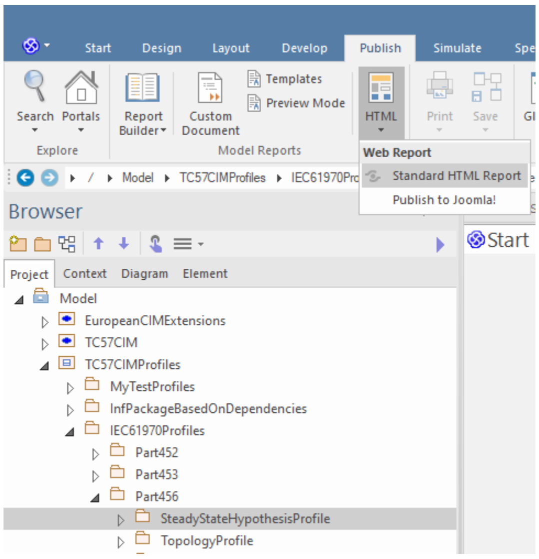
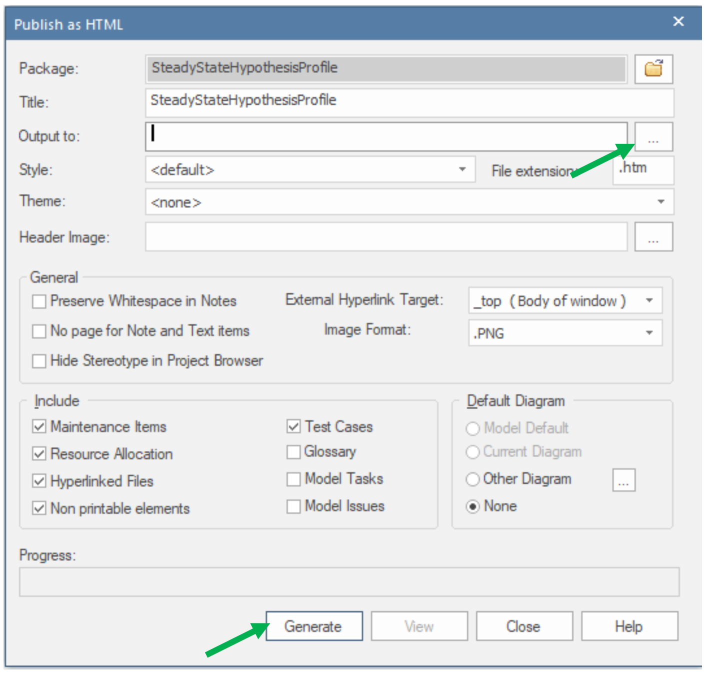


Figure 90 – Launch of HTML export in Enterprise Architect

The dialog shown in Figure 91 will appear. Select button "…" to set the location and the name. Then select button "Generate".



Publish as HTML

Package:	SteadyStateHypothesisProfile
Title:	SteadyStateHypothesisProfile
Output to:	
Style:	<default>
File extension:	.htm
Theme:	<none>
Header Image:	

General

- [ ] Preserve Whitespace in Notes
- [ ] No page for Note and Text items
- [ ] Hide Stereotype in Project Browser

External Hyperlink Target: _top (Body of window)

Image Format: .PNG

Include

- [x] Maintenance Items
- [x] Resource Allocation
- [x] Hyperlinked Files
- [x] Non printable elements
- [x] Test Cases
- [ ] Glossary
- [ ] Model Tasks
- [ ] Model Issues

Default Diagram

Model Default / Current Diagram / Other Diagram [✓ None]

Progress:

Figure 91 – “Publish as HTML” dialog of Enterprise Architect

Bibliography

[1] CimConteXtor Manual, 2019: https://www.cimcontextor.net/index.php/adwls

[2] CimSyntaxGen Manual 2018: https://www.cimcontextor.net/index.php/adwls

[3] jCleanCim: http://www.tanjakostic.org/jcleancim/

[4] CIMug, CIM Modelling Guide:

https://cimug.ucaiug.org/Model%20Manager%20Documents/Public/CIM%20Modeling%20Guide_v1.1.pdf

Annexes

11.1 Annex A: Example of CimSyntaxGen configuration file for CGMES v2.4

```
<?xml version="1.0"?>
<add-in>
  <appSettings>
    <configuration name="VersionConfig" value="2020-11-03" />
    <configuration name="Log" value="Checked" />
    <configuration name="StereotypeNameSpaces" value="Checked" />
    <configuration name="Xmlbase" value="Checked" />
    <configuration name="ImportCodeList" value="Checked" />
    <configuration name="Esmp" value="Checked" />
    <configuration name="ModeBatch" value="Unchecked" />
    <configuration name="ModeTest" value="False" />
    <configuration name="EntsoeKeepInheritancePath" value="Checked" />
    <configuration name="EntsoeKeepUMLDiagrams" value="Checked" />
    <configuration name="EntsoeKeepClassDiagrams" value="Checked" />
    <configuration name="EntsoeKeepHyperlinks" value="Checked" />
    <configuration name="EntsoeManageCodeList" value="Checked" />
    <configuration name="jsonEnabled" value="Checked" />
    <configuration name="NavigationEnabled" value="Unchecked" />
    <configuration name="CodeComponent" value="Unchecked" />
    <configuration name="XSDByRef" value="Unchecked" />
    <configuration name="SelectedCanonicalPackagesForXsdToProf" value="IEC61970,IEC61968" />
    <configuration name="ImageExt" value=".png" />// can be .emf, bmp, .jpg, .gif, .png, .tga
  </appSettings>
  <dataProfile>
    <profdata name="DefaultXSDFilePath" value=".\\essai.xsd" />
    <profdata name="CurrentXSDFilePath" value="C:\Users\andell3\Documents\essais\json\EndDeviceEvents.json" />
    <profdata name="ExtensiontXSDFilePath" value=".\\urn-entsoe-eu-local-extension-types.xsd" />
    <profdata name="DefaultProfileNamespace=" value="http://iec.ch.tc57/Profile1#" />
    <profdata name="DefaultTargetNamespace" value="urn:IEC62325.351:wg16:acknowledgementdocument:5:1" />
    <profdata name="ProfileNamespace" value="http://iec.ch/TC57/ns/CIM/DiagramLayout-EU/3.0#" />
    <profdata name="TargetNamespace" value="urn:ebix.eu:ProofOfConcept:1:0" />
    <profdata name="DefaultRootModelURI" value="http://iec.ch/TC57/2013/CIM-schema-cim16#" />
    <profdata name="RootModelURI" value="http://iec.ch/TC57/2013/CIM-schema-cim16#" />
    <profdata name="Prefix" value="cim" />
    <profdata name="EnvelopName" value="MyEvents" />
    <profdata name="SelectedProfileForJson" value="680" />
    <profdata name="URICodelist" value="urn:entsoe.eu:wgedi:codelists" />
    <profdata name="CodeListLocation" value="urn:entsoe.eu:wgedi:codelists" />
    <profdata name="CodeListLocalLocation" value="urn-entsoe-eu-local-extension-types.xsd" />
    <profdata name="URISchemaLocation" value="" />
    <profdata name="PrefixCodelist" value="ecl" />
    <profdata name="Version" value="1.0" />
    <profdata name="ListStereoNamespace" value="Entsoe|Operation|Abstract|ShortCircuit|Description" />
    <profdata name="ListStereoNamspaceExportable" value="ShortCircuit|Operation" />
    <profdata name="EntsoeEquipmentProfile" value="EquipmentProfile" />
    <!--    " DataTypesDomain  used for certain profile architecture when such a domain exists " -->
    <profdata name="DataTypesDomain" value="DomainProfile" />
    <profdata name="ListSingularName" value="Status|Address|StreetAddress" />
    <profdata name="EntsoeExpStereo" value="Entsoe" />
    <!--    "PackageExtension        deprecated will be deleted in future version " -->
    <profdata name="PackageExtension" value="Extension" />
    <profdata name="LastExportedProfile" value="C:\Users\andell3\Documents\essais\chavdar\eserdf2.rdf" />
    <profdata name="FlatDatatypes" value="" />
    <profdata name="CCTypeOfDocument" value="IS" />
    <profdata name="CCStandardNumber" value="61970-600 ed1" />
    <profdata name="CCProfileName" value="DiagramLayoutProfile" />
    <profdata name="CCCodeComponentName" value="XSD_Schema" />
    <profdata name="CCDiffusionState" value="Draft" />
    <profdata name="CCVersionStateInfo" value="3.0.0" />
    <profdata name="CCselectedProfiles" value="" />
    <profdata name="CCLightFull" value="full" />
    <profdata name="CCPublicationDate" value="2020-10-12" />
    <profdata name="CCSelectedFiles" value="" />
    <profdata name="CCPackageName" value="" />
    <shacldata name="imports" value="" />
    <shacldata name="descriptionAttributeCardinality" value="This constraint validates the cardinality of the property
(attribute)." />
    <shacldata name="messageAttributeCardinality" value="Cardinality violation. Upper bound shall be 1 | Cardinality
violation. Missing required property (attribute). | Cardinality violation. Upper bound shall be 1 or Missing required property
(attribute) " />
    <shacldata name="descriptionAssociationCardinality" value="This constraint validates the cardinality of the association
at the used direction." />
```

```
    <shacldata name="messageAssociationCardinality" value="Cardinality violation. Upper bound shall be 1. | Cardinality
violation. Missing required association.| Cardinality violation. Upper bound shall be 1 or Missing required association." />
    <shacldata name="descriptionValueType" value="This constraint validates the value type of the association at the used
direction." />
    <shacldata name="messageValueType" value="One of the following does not conform: 1) The value type shall be IRI; 2)
The value type shall be an instance of the class: {the IRI of the class}." />
    <shacldata name="descriptionDatatype" value="This constraint validates the datatype of the attribute" />
    <shacldata name="messageDatatype" value="The datatype is not literal or it violates the xsd datatype. | The datatype is
not IRI (Internationalized Resource Identifier) or it is an enumerated value which is not part of the enumeration." />
    <profstereo name="Entsoe" prefix="entsoe" uri="http://entsoe.eu/CIM/SchemaExtension/3/1#" html="Checked"
rdfs="Checked" desc="Checked" />
    <profstereo name="Operation" prefix="cim" uri="http://iec.ch/TC57/2013/CIM-schema-cim16#" html="Checked"
rdfs="Checked" desc="Checked" />
    <profstereo name="Abstract" prefix="cim" uri="http://iec.ch/TC57/2013/CIM-schema-cim16#" html="Checked"
rdfs="Checked" desc="Checked" />
    <profstereo name="ShortCircuit" prefix="cim" uri="http://iec.ch/TC57/2013/CIM-schema-cim16#" html="Checked"
rdfs="Checked" desc="Checked" />
          <profstereo name="Description" prefix="cim" uri="http://iec.ch/TC57/2013/CIM-schema-cim16#" html="Checked"
rdfs="Checked" desc="Checked" />
  </dataProfile>
</add-in>
```

11.2 Annex B: Example of CimSyntaxGen configuration file for CGMES v3.0

```
<?xml version="1.0"?>
<add-in>
  <appSettings>
    <configuration name="VersionConfig" value="2020-11-03" />
    <configuration name="Log" value="Checked" />
    <configuration name="StereotypeNameSpaces" value="Checked" />
    <configuration name="Xmlbase" value="Checked" />
    <configuration name="ImportCodeList" value="Checked" />
    <configuration name="Esmp" value="Checked" />
    <configuration name="ModeBatch" value="Unchecked" />
    <configuration name="ModeTest" value="False" />
    <configuration name="EntsoeKeepInheritancePath" value="Checked" />
    <configuration name="EntsoeKeepUMLDiagrams" value="Checked" />
    <configuration name="EntsoeKeepClassDiagrams" value="Checked" />
    <configuration name="EntsoeKeepHyperlinks" value="Checked" />
    <configuration name="EntsoeManageCodeList" value="Checked" />
    <configuration name="jsonEnabled" value="Checked" />
    <configuration name="NavigationEnabled" value="Unchecked" />
    <configuration name="CodeComponent" value="Unchecked" />
    <configuration name="XSDByRef" value="Unchecked" />
    <configuration name="SelectedCanonicalPackagesForXsdToProf" value="IEC61970,IEC61968" />
    <configuration name="ImageExt" value=".png" />// can be .emf, bmp, .jpg, .gif, .png, .tga
  </appSettings>
  <dataProfile>
    <profdata name="DefaultXSDFilePath" value=".\\essai.xsd" />
    <profdata name="CurrentXSDFilePath" value="C:\Users\andell3\Documents\essais\json\EndDeviceEvents.json" />
    <profdata name="ExtensiontXSDFilePath" value=".\\urn-entsoe-eu-local-extension-types.xsd" />
    <profdata name="DefaultProfileNamespace=" value="http://iec.ch.tc57/Profile1#" />
    <profdata name="DefaultTargetNamespace" value="urn:IEC62325.351:wg16:acknowledgementdocument:5:1" />
    <profdata name="ProfileNamespace" value="http://iec.ch/TC57/ns/CIM/DiagramLayout-EU/3.0#" />
    <profdata name="TargetNamespace" value="urn:ebix.eu:ProofOfConcept:1:0" />
    <profdata name="DefaultRootModelURI" value="http://iec.ch/TC57/CIM100#" />
    <profdata name="RootModelURI" value="http://iec.ch/TC57/CIM100#" />
    <profdata name="Prefix" value="dl" />
    <profdata name="EnvelopName" value="MyEvents" />
    <profdata name="SelectedProfileForJson" value="680" />
    <profdata name="URICodelist" value="urn:entsoe.eu:wgedi:codelists" />
    <profdata name="CodeListLocation" value="urn:entsoe.eu:wgedi:codelists" />
    <profdata name="CodeListLocalLocation" value="urn-entsoe-eu-local-extension-types.xsd" />
    <profdata name="URISchemaLocation" value="" />
    <profdata name="PrefixCodelist" value="ecl" />
    <profdata name="Version" value="1.0" />
    <profdata name="ListStereoNamespace" value="European|Description" />
    <profdata name="ListStereoNamspaceExportable" value="ShortCircuit|Operation" />
    <profdata name="EntsoeEquipmentProfile" value="EquipmentProfile" />
    <!--    " DataTypesDomain  used for certain profile architecture when such a domain exists " -->
    <profdata name="DataTypesDomain" value="DomainProfile" />
    <profdata name="ListSingularName" value="Status|Address|StreetAddress" />
    <profdata name="EntsoeExpStereo" value="European" />
    <!--    "PackageExtension        deprecated will be deleted in future version " -->
    <profdata name="PackageExtension" value=" EuropeanCIMExtensions " />
    <profdata name="LastExportedProfile" value="C:\Users\andell3\Documents\essais\chavdar\eserdf2.rdf" />
    <profdata name="FlatDatatypes" value="" />
    <profdata name="CCTypeOfDocument" value="IS" />
    <profdata name="CCStandardNumber" value="61970-600 ed1" />
    <profdata name="CCProfileName" value="DiagramLayoutProfile" />
    <profdata name="CCCodeComponentName" value="XSD_Schema" />
    <profdata name="CCDiffusionState" value="Draft" />
    <profdata name="CCVersionStateInfo" value="3.0.0" />
    <profdata name="CCselectedProfiles" value="" />
    <profdata name="CCLightFull" value="full" />
    <profdata name="CCPublicationDate" value="2020-10-12" />
    <profdata name="CCSelectedFiles" value="" />
    <profdata name="CCPackageName" value="" />
    <shacldata name="imports" value="" />
    <shacldata name="descriptionAttributeCardinality" value="This constraint validates the cardinality of the property
(attribute)." />
    <shacldata name="messageAttributeCardinality" value="Cardinality violation. Upper bound shall be 1 | Cardinality
violation. Missing required property (attribute). | Cardinality violation. Upper bound shall be 1 or Missing required property
(attribute) " />
    <shacldata name="descriptionAssociationCardinality" value="This constraint validates the cardinality of the association
at the used direction." />
    <shacldata name="messageAssociationCardinality" value="Cardinality violation. Upper bound shall be 1. | Cardinality
violation. Missing required association.| Cardinality violation. Upper bound shall be 1 or Missing required association." />
    <shacldata name="descriptionValueType" value="This constraint validates the value type of the association at the used
direction." />
    <shacldata name="messageValueType" value="One of the following does not conform: 1) The value type shall be IRI; 2)
The value type shall be an instance of the class: {the IRI of the class}." />
    <shacldata name="descriptionDatatype" value="This constraint validates the datatype of the attribute" />
    <shacldata name="messageDatatype" value="The datatype is not literal or it violates the xsd datatype. | The datatype is
not IRI (Internationalized Resource Identifier) or it is an enumerated value which is not part of the enumeration." />
    <profstereo name="European" prefix="eu" uri="http://iec.ch/TC57/CIM100-European#" html="Checked" rdfs="Checked"
desc="Checked" />
         <profstereo name="Description" prefix="cim" uri="http://iec.ch/TC57/CIM100#" html="Checked" rdfs="Checked"
desc="Checked" />
  </dataProfile>
</add-in>
```