jCleanCim introduction

January 2018 (02v02)

tatjana.kostic@ieee.org

About jCleanCim

jCleanCim is an open source tool:

- since 02v00 provided under terms of GNU LGPLv3 license
- http://www.tanjakostic.org/jcleancim

Developed to support validation and documentation generation from Enterprise Architect CIM and IEC 61850 UML models.

A Java application, but (for some tasks) platform dependent due to usage of applications available on MS Windows only:

- Enterprise Architect
- MS Word

A console application, currently without any GUI.

Use the latest version available.

Who should use jCleanCim?

Primarily those who edit CIM or IEC61850 UML and publish its documentation, thus:

- Official IEC CIM model editors, responsible for maintaining the CIM information model (UML) and for generating official IEC documents, and,
- Official IEC 61850 model managers, responsible for maintaining the IEC 61850 UML model and for generating official IEC documents, and,
- Official IEC CIM profile document editors, if their profiles are available in UML, for generating official IEC documents, and,
- Those who define custom (non-standard) CIM or IEC 61850 extensions who want to ensure they have followed standard UML modelling rules and who want to generate documentation for those extensions.

When should you use jCleanCim?

After editing CIM or IEC61850 UML, to validate the edits (reinforce rules).

When you need to collect the numbers (model statistics).

To produce MS Word documentation from UML models.

To produce Web Access XML from UML models.

If you are a CIMTool user:

- You would first use jCleanCim to validate correctness of the CIM information model (UML) / IEC 61850 UML model, and if required, to generate the information model documentation in MS Word or XML format, as required by IEC process.
- You would then use CIMTool to create CIM profiles (XSD, RDFS, OWL) and their documentation (HTML) from the imported CIM UML model, and to validate instance files created based on those profiles – independently of jCleanCim.

If you are a UML-based profiling tools user:

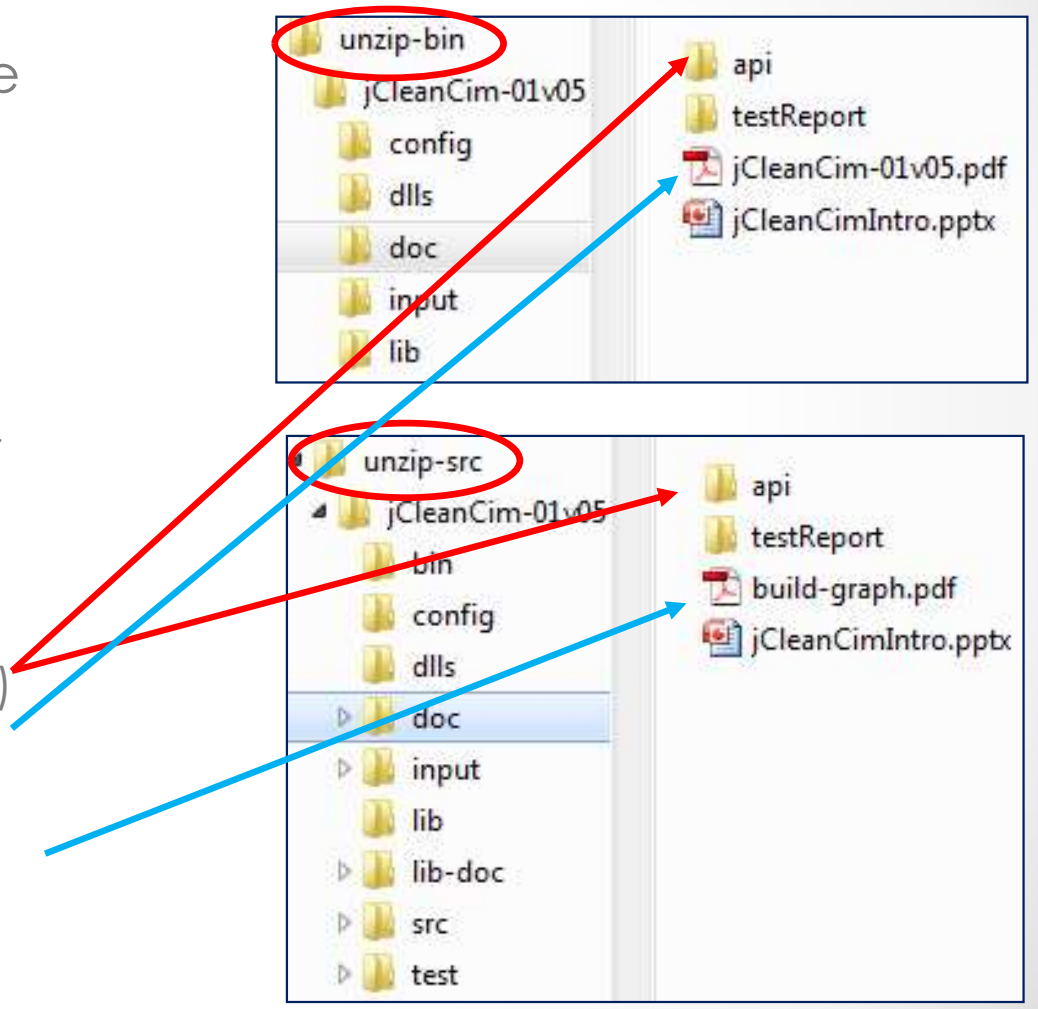
You are probably already using jCleanCim to generate MS Word documentation.

Documentation

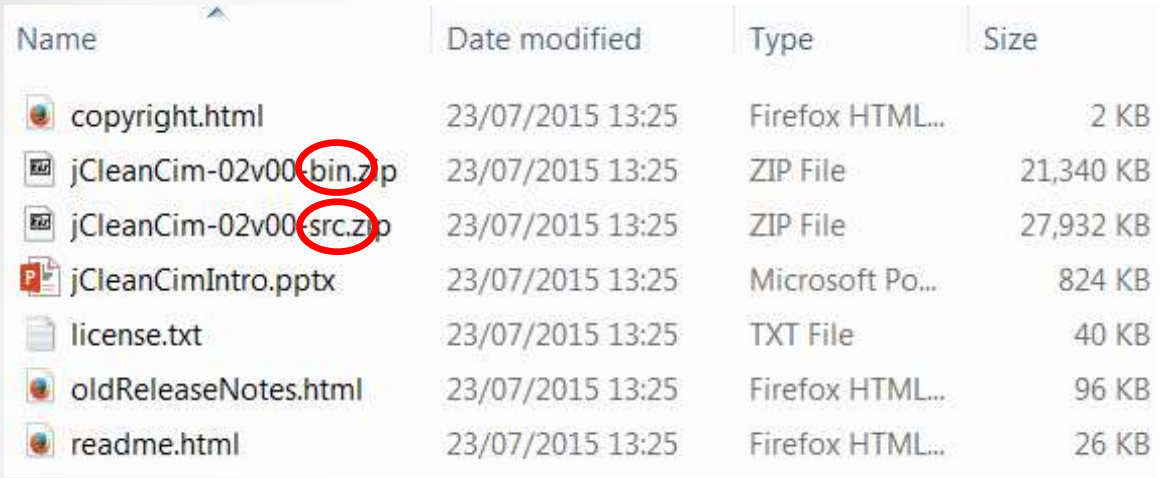
Start from this presentation and the readme.html, available on-line and also bundled with every distribution.

Once you unzip a jCleanCim distribution, detailed documentation is available under the project's .doc directory:

- As javadoc in doc/api/index.html
- For binary distribution, also as .pdf (auto-generated from the javadoc)
- For source distribution, also build targets dependencies



Selecting distribution



Name	Date modified	Type	Size
copyright.html	23/07/2015 13:25	Firefox HTML...	2 KB
jCleanCim-02v00-bin.zip	23/07/2015 13:25	ZIP File	21,340 KB
jCleanCim-02v00-src.zip	23/07/2015 13:25	ZIP File	27,932 KB
jCleanCimIntro.pptx	23/07/2015 13:25	Microsoft Po...	824 KB
license.txt	23/07/2015 13:25	TXT File	40 KB
oldReleaseNotes.html	23/07/2015 13:25	Firefox HTML...	96 KB
readme.html	23/07/2015 13:25	Firefox HTML...	26 KB

bin distribution

- for jCleanCim end-user

src distribution

- for jCleanCim developer, packager and end-user, with Apache ant
- contains also eclipse project files (unzip, then import -> existing project)

Note for users with 64-bit Windows 7

If you have a 64-bit Windows OS:

- ensure you install a 32-bit Java runtime (JRE) if you run a binary distribution, or Java SDK (software development kit) if you run a source distribution
- ensure you have that 32-bit Java appear on your PATH before potentially already installed 64-bit Java

Reason:

- Enterprise Architect is still a 32-bit application and requires a 32-bit Java

One possible fix, lasting until next reboot:

- See the commented text in the run.bat script in jCleanCim distribution
- Uncomment this line, by removing the initial "rem"

```
rem set PATH=C:\Program Files (x86)\Java\Jre7\bin;%PATH%
```

- This will put your 32-bit Java runtime before a potential 64-bit installation

Note for CIMTool users (1/2)

In contrast to CIMTool, which is an eclipse-based application with a GUI:

- jCleanCim only uses eclipse for development and compilation
- jCleanCim is a simple console application, without a GUI

The most comfortable way to use jCleanCim is however with *-src.zip distribution in eclipse, because:

- You click to run jCleanCim instead of typing commands in the console window
- Eclipse gives a nice console output (you can copy/paste/search/scroll easily)
- Since 02v00, you'll need to import not eclipse project archive, but simply an existing eclipse project (unzipped directory)

If you are developing in Java with eclipse, you already have what is needed.

Note for CIMTool users (2/2)

If you have CIMTool that contains an eclipse runtime:

- Using that installation of eclipse (runtime) is **not** sufficient, because it is only runtime, without support for Java code development
- You must have an SDK (software development kit) to automatically build the jCleanCim application from sources

On eclipse download site, the minimum required distribution is "Eclipse IDE for Java developers"

- you can then install CIMTool plug-in in this (or more recent) version of eclipse

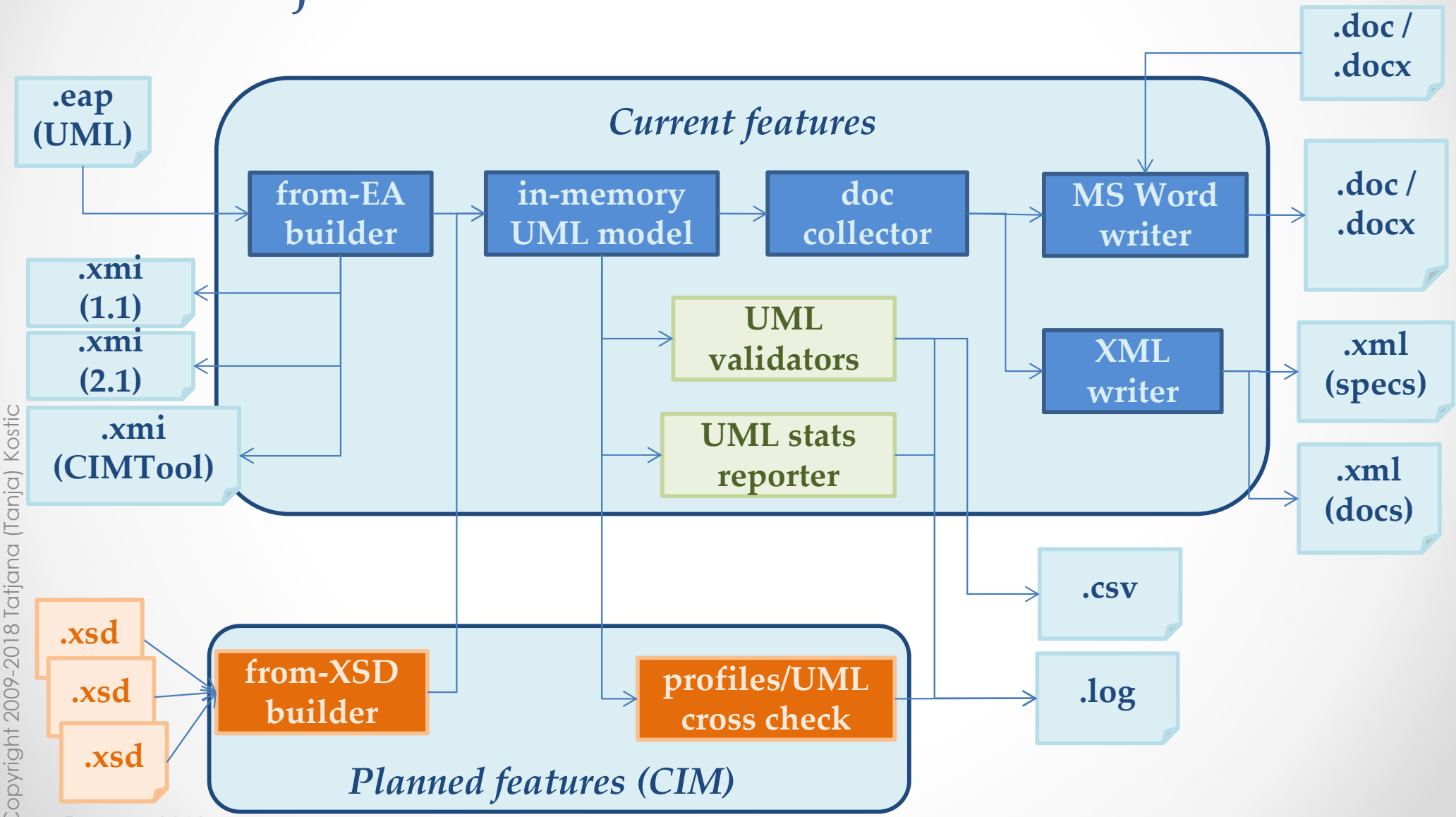
Features & configuration overview

• • •

UML model export to XMI
UML model validation
UML model statistics
MS Word doc generation from UML (and from CIM profiles*)
XML doc generation from UML
CIM profiles vs. UML model cross-check*

*being implemented

jCleanCim features overview



jCleanCim features intro (1/2)

jCleanCim **first**:

- Creates in-memory representation of the whole content of UML from .eap file
- (if set in properties – see next slide) Can **export the model to the three XMI formats**
- Can **selectively export one or more XMI formats**

On the fly

- It analyses the model and calculates a bunch of things, including effective dependencies
- It logs that all into log files under **log** directory, automatically created on the first run

jCleanCim features intro (2/2)

After that, depending on what is set in properties (see next slide), one or more of the following gets executed and logged:

- **Validation** of a UML model provided in an .eap file: UML of standard IEC CIM (base and extensions), UML of IEC61850 family, and custom extensions of any of these
- Calculation and printing of **statistics** of the UML model
- **Generation of MS Word documentation** from the UML model
- **Generation of XML documentation** from the UML model
- (CIM only, *being implemented*) **Generation of MS Word documentation** from CIMTool .xsd profiles
- (CIM only, *being implemented*) **Cross-check of profiles** against the CIM UML model.

jCleanCim configuration (1/2)

Use file **config/config.properties.**

Minimum configuration for CIM* **validation and stats**:

```
model.filename    = base-small.eap

validation.on     = true
statistics.on     = true
```

Minimum configuration for automatic **XMI export** (used by CIM model managers):

```
model.filename    = base-small.eap
model.builder     = sqlxml

xmiexport.on      = true
```

You can define a number of <custom-name>.properties files and run any one of them with command line argument:

$ run –propFile <custom-name>.properties

* UML of IEC61850 needs more than this, see config61850.properties and doc in Configuration class

jCleanCim configuration (2/2)

Minimum configuration for CIM* **MS Word doc generation**:

```
model.filename                  = base-small.eap
model.builder                   = sqlxml

docgen.on                       = true
profiles.docgen.on              = false

**  docgen.word.useDocFormat    = true
    docgen.word.saveReopenEvery = 12
    docgen.word.inTemplate      = base-small-template.doc
    docgen.word.outDocument     = base-small.doc
```

Minimum configuration for CIM* **XML doc generation**:

```
model.filename      = base-small.eap
model. builder      = sqlxml

docgen.on           = true
profiles.docgen.on  = false

docgen.xml.outSpec = base-small-tool01v06-spec.xml
docgen.xml.outDoc = base-small-tool01v06-doc.xml
```

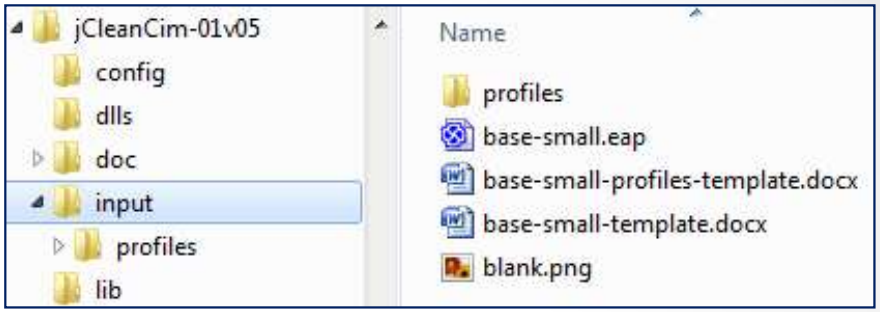
* UML of IEC61850 needs more than this, see config61850.properties and doc in Configuration class

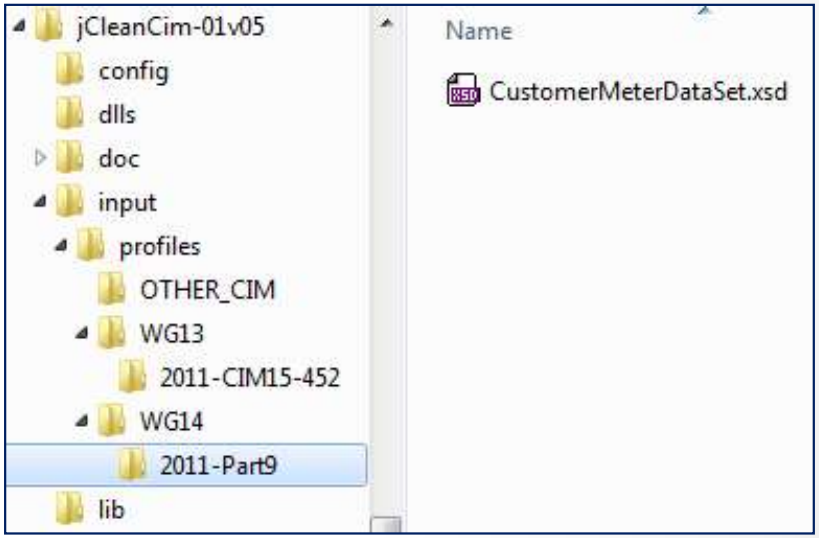
** Preparing for faster implementation based on OpenXML (.docx)

jCleanCim input files

For any function, you need at least an .eap model file

- for MS Word document generation, you also need an MS Word template (regular .docx file) containing particular jCleanCim placeholders
- *(CIM only, not implemented yet)* for profile cross-check with the UML model, you also need one or more profiles with XSD syntax, generated by CIMTool





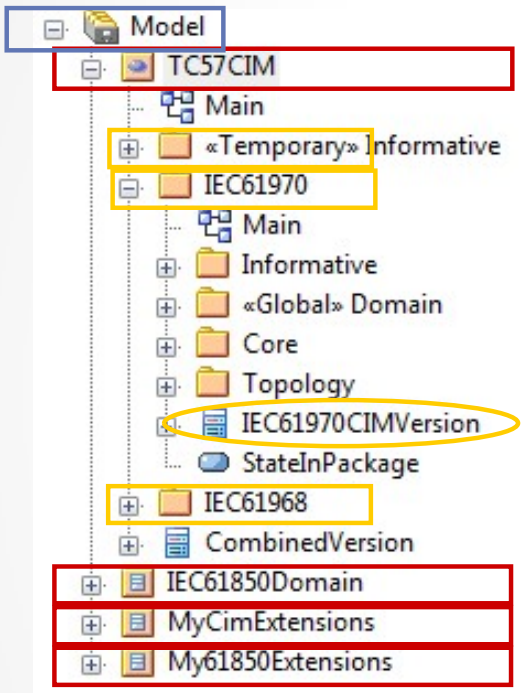
All distributions contain sets of files in the project's **input** directory:

- base-small.eap – small subset of base CIM and IEC 61850 with **lots of buggy constructs**, on purpose, to show DON'T's
- sub-directories within input/profiles contain trimmed samples of CPSM and of one metering profile

Copy your own .eap and .doc/.docx files to the project's input directory.

(CIM only) Copy your .xsd files anywhere below the project's input/profiles directory.

Recommended UML model structure (1/2)

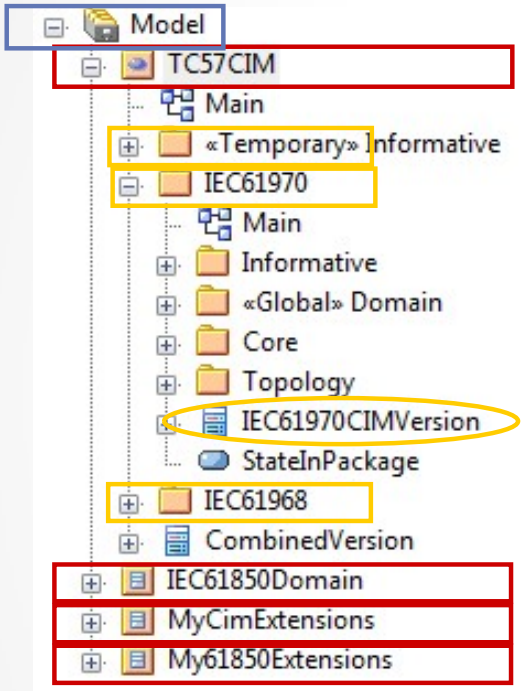


(example from base-small.eap, does not reflect the full model, just small part of it)

- One root (here: "Model")
  - Currently, jCleanCim ignores everything but the first root in the .eap project
- Any number of **model packages** under the root
  - Each with ***either*** CIM (default) or IEC61850 nature
    - IEC61850 nature must be explicitly specified in config61850.properties file, with **model.nature.iec61850** property
- Any number of **top-level packages** (per WG owner) under the model package
  - A top-level package expected to contain a (UML) version class with correct name

model.nature.iec61850	= IEC61850Domain, My61850Extensions

Recommended UML model structure (2/2)

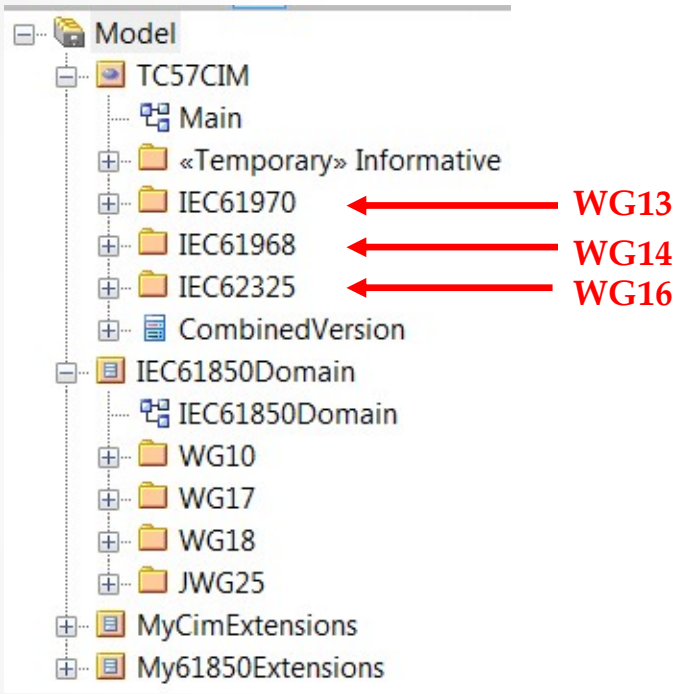


(example from base-small.eap, does not reflect the full model, just small part of it)

Rationale

- Preserves the current standard CIM "place" in the .eap project
- Clearly separates CIM and non-CIM models
  - Nature need not be encoded in UML, but in properties file
- Clearly separates standard UML and non-standard extensions
  - Easy to evolve/update/merge standard model as it evolves
  - Easy to independently evolve/update/merge variations of custom extensions, without interfering with the standard UML
- Top-level packages may be associated with IEC TC57 WGs (or to projects, for custom extensions)

Standard UML top-package owners



(example from base-small.eap, does not reflect the full model, just small part of it)

Currently, jCleanCim encodes the mapping of top-level packages and IEC owners:

- CIM owners are WG13, WG14, WG16
- IEC61850 owners are WG10, WG17, WG18, JWG25 (and WG19)

All other UML packages and elements get assigned the owner "OTHER_CIM" or "OTHER_IEC61850".

Features:
Intro - model building

• • •

Or

In eternal quest for speeding up the slow EA API implementation

Reading model from .eap repository:

Minimum configuration*

For any feature relying on .eap UML model, you must specify the UML model file name.

Copy your own model file(s) into the project's **input** directory.

Since 01v08, we have 3 implementations – see next slide for comparison.

- Default: Use if you ***don't*** need diagram or XMI export. Useful also for doc-generation ***without*** diagrams.
- Use for preparing a release (XMI export) and for full document generation.
- Avoid !(kept as a fallback in case EA changes its internal DB schema).

```
model.filename   = base-small.eap
```

OR

```
model.filename   = base-small.eap
model.builder    = db
```

```
model.filename   = base-small.eap
model.builder    = sqlxml
```

```
model.filename   = base-small.eap
model.builder    = japi
```

* UML of IEC61850 needs more than this, see config61850.properties and doc in Configuration class

Reading model from .eap repository:

Builder comparisons

model.builder=	db	sqlxml	japi
how it reads .eap model file	as Access DB	EA Java API: queries (SQL) + result-set (XML)	EA Java API: iterating content
speed: iterating model	as fast as it gets	pretty fast	extremely slow
speed: opening .eap file		very slow	
needs ea.jar + ea.dll	no	yes	
bound to M$ Windows	no	yes	
can export UML diagrams	never	yes (if docgen.on=true)	
can export XMI	never	yes (if xmiexport.on=true)	

Features:
UML model export to XMI

• • •

Useful for CIM model managers

XMI export:

Minimum configuration*

You must specify the UML model file name, a builder that uses EA repository API, and enable XMI export.

Copy your own model file(s) into the project's **input** directory.

This will export all the supported formats:

- except for 'cimtool' - because CIMTool now (since 1.9.6) can import .eap and does not need .xmi, which is 82 MB for latest CIM !

```
model.filename   = base-small.eap
model.builder    = sqlxml

xmiexport.on     = true
```

* UML of IEC61850 needs more than this, see config61850.properties and doc in Configuration class

XMI export:

Overview

CIM model managers need to export UML model to several XMI formats and package those .xmi files into release

- Manually, it's a tedious, error-prone and time-consuming process
- jCleanCim can do that automatically

We take the model file name, and replace its .eap extension appropriately

- exported files go to the project's **output** directory

IEC61850 model managers don't need this functionality (yet).

XMI export:

Fine tuning

- This will export only XMI appropriate for CIMTool (Rose UML 1.4, without diagrams)

```
model.filename      = base-small.eap
model.builder       = sqlxml

xmiexport.on        = true
xmiexport.dialects  = cimtool
```

- This will export all 3 currently supported formats (two defaults, plus 'cimtool')

```
model.filename      = base-small.eap
model.builder       = sqlxml

xmiexport.on        = true
xmiexport.dialects  = ea_xmi11, ea_xmi21, cimtool
```

Features:
UML model validation

• • •

Compiler is my friend.

Validation:

Minimum configuration*

You must specify the UML model file name and enable validation.

Copy your own model file(s) into the project's **input** directory.

```
model.filename   = base-small.eap

validation.on    = true
```

* UML of IEC61850 needs more than this, see config61850.properties and doc in Configuration class

Validation:

Overview (1/2)

Validators for 7 kinds of UML elements

- Associations
- Attributes
- Classes
- Packages
- Diagrams
- Dependencies (hand-drawn in UML), and
- Operations (for UML of IEC61850 only)

Validators deal with rules:

- Model consistency
- CIM/IEC61850 UML naming and design rules
- Potential model editor errors
- Identifying remains from Rose or other imported models
- Illegal/redundant UML constructs (that EA sometimes allows…)

Validation:

Overview (2/2)

Each rule is a class

- Inheriting from a common abstract class, and implementing an interface
- Javadoc in package **org.iec.tc57.jcleancim.validation** provides guidelines on how to add a rule to a validator
- It is easy to add a new rule on need!

Full list of rules currently available is on the next two slides, with a couple of annotated examples for class validator rules

- Some rules apply in general, some apply to CIM models only, and some to IEC61850 models only
- List is produced through log (also CIM-specific or 61850-specific)
- Javadoc also contains UML for all the classes (not shown below)

Validation rules:

Classes

Available rules in ClassValidator (category, severity):

- CimClassesWithUnexpectedElements (permissiveTool, high)
- ClassesWithUnexpectedConnectors (permissiveTool, high)
- EnumClassesWithNoLiterals (modellingRule, high)
- CimCompoundClassesWithNoAttributes (modellingRule, high)
- EnumClassesWithSingleLiteral (modellingRule, medium)
- EnumClassesWithTwoLiterals (modellingRule, low)
- EnumClassesWithBadName (modellingRule, high)
- CimPrimitiveClassesWithAttributes (modellingRule, high)
- CimPrimitiveClassesWithIllegalOwner (modellingRule, high)
- ClassesWithDuplicateInheritedAttributeNames (modellingRule, high)
- ClassesWithDuplicateOwnOrInheritedAssociationEndNames (modellingRule, high)
- ClassesWithSelfInheritance (permissiveTool, high)
- ClassesWithSelfDependency (permissiveTool, high)
- ClassesWithLeafPropSet (modellingRule, high)
- ClassesWithRootPropSet (modellingRule, high)
- ClassesWithPersistentPropSet (modellingRule, high)
- ClassesWithMultipleSuperclasses (modellingRule, high)
- ClassesWithSuperclassesFromUnallowedPackage (modellingRule, high)
- ClassesThatShouldNotBeAssociationClass (modellingRule, high)
- ClassesWithUnallowedStereotype (modellingRule, high)
- CimClassesWithOldDatatypeStereotype (modellingRule, medium)
- CimClassesUsedForAttributesButHaveAssociations (modellingRule, high)
- CimClassesUsedForAttributesButHaveSubclasses (modellingRule, high)
- CimClassesUsedForAttributesButHaveSuperclasses (modellingRule, high)
- CimClassesThatShouldNotBeAbstract (modellingRule, high)
- CimClassesThatShouldNotHaveOperations (modellingRule, high)
- CimClassesThatShouldNotHaveExplicitDependencies (modellingRule, high)
- ClassesThatShouldNotHaveNestingThroughAttribute (modellingRule, high)
- Iec61850ClassesThatShouldHaveAliasAsTitle (modellingRule, high)
- Iec61850ClassesThatShouldHaveTaggedValuesForDocgen (modellingRule, high)
- CimClassesNeverUsedInRelationships (modellingRule, high)
- ClassesWithUnallowedTagNames (modellingRule, high)
- Iec61850ClassesWithInvalidConstraints (modellingRule, high)
- Iec61850LNClassesWithSuperfluousConstraints (modellingRule, high)
- Iec61850ClassesWithMissingCondIDTextInConstraints (modellingRule, high)
- CimDatatypeClassesWithInvalidAttributes (modellingRule, high)
- ClassesMissingDoc (documentationRule, medium)
- ClassesWithBadDocStart (documentationRule, low)
- ClassesWithBadDocEnd (documentationRule, low)
- ClassesWithBadCharacterInName (namingRule, high)
- CimClassesNameStartingWithLowerCase (namingRule, high)
- CimClassesNameShouldBeSingular (namingRule, high)
- Iec61850LNClassesInWrongGroup (namingRule, medium)
- Iec61850LNClassesMalformedName (namingRule, high)
- EnumClassesWithSomeCodesMissing (modellingRule, high)
- EnumClassesWithDuplicateCodes (modellingRule, high)
- ClassesWithSameName (modellingRule, high)
- CimClassesNeverUsedAsTypeForAttribute (modellingRule, high)

CIM rules

Remains from Rose

EA allows for this

Style warning

Model inconsis tencies

Editor's error

Validation rules:

Dependencies (hand-drawn), diagrams, operations, packages, associations, attributes

Available rules in PackageValidator (category, severity):
- PackageUnexpectedElements (modellingRule, medium)
- PackageUnexpectedConnectors (modellingRule, medium)
- PackagesWithSelfDependency (permissiveTool, high)
- PackagesWithUnallowedStereotype (modellingRule, high)
- PackagesTopLevelWithoutVersionClass (modellingRule, high)
- Iec61850PackagesThatShouldHaveAliasAsTitle (modellingRule, high)
- PackagesWithUnallowedTagNames (modellingRule, high)
- PackagesMissingDoc (documentationRule, medium)
- PackagesWithBadDocStart (documentationRule, low)
- PackagesWithBadDocEnd (documentationRule, low)
- PackagesWithBadCharacterInName (namingRule, high)
- PackagesWithSameName (modellingRule, high)

Available rules in AttributeValidator (category, severity):
- EnumLiteralsWithSuperfluousType (modellingRule, high)
- EnumLiteralsWithInitValue (modellingRule, high)
- EnumLiteralsWithoutEnumStereotype (modellingRule, high)
- AttributesWithInvalidMultiplicity (modellingRule, high)
- CimAttributesThatShouldBeOptional (modellingRule, high)
- AttributesWithInvalidTypeNull (modellingRule, high)
- AttributesWithInvalidTypeString (modellingRule, high)
- AttributesWithTypeIdMismatch (modellingRule, high)
- CimAttributesThatShouldBePublic (modellingRule, high)
- AttributesThatAreStaticButNotConst (modellingRule, high)
- CimAttributesThatAreNotStaticNonConstWithInitVal (modellingRule, high)
- AttributesThatAreConstNonStatic (modellingRule, high)
- AttributesWithUnallowedStereotype (modellingRule, high)
- AttributesThatAreEnumsInNonEnumeratedClass (modellingRule, high)
- CimAttributesThatShouldBeReplacedWithAssociation (modellingRule, high)
- AttributesWhoseTypeIsInformative (modellingRule, high)
- AttributesWithUnallowedTagNames (modellingRule, high)
- Iec61850AttributesWithInexistingSibling (modellingRule, high)
- CimAttributesWithFlagInName (namingRule, medium)
- AttributesMissingDoc (documentationRule, medium)
- AttributesWithBadDocStart (documentationRule, low)
- AttributesWithBadDocEnd (documentationRule, low)
- CimAttributesWithBadCharacterInName (namingRule, high)
- Iec61850AttributesWithBadCharacterInName (namingRule, high)
- Iec61850DOAttributesWithTooLongName (namingRule, high)
- Iec61850FCDAAttributesWithMissingConstraint (modellingRule, high)
- AttributesWithInexistingEnumLiteralAsInitValue (permissiveTool, high)
- Iec61850DOAttributesWithNameMissingAbbreviation (modellingRule, high)
- CimAttributesNameStartingWithUpperCase (namingRule, high)
- CimAttributesNameShouldBeSingular (namingRule, high)
- CimAttributesNameShouldNotStartWithClassName (namingRule, medium)
- Iec61850AbbreviationLiteralsNameStartingWithLowerCase (namingRule, high)
- Iec61850DOAttributesNameStartingWithLowerCase (namingRule, high)
- Iec61850DOAbbreviationLiteralsDuplicateName (modellingRule, high)
- Iec61850DOAbbreviationLiteralsDuplicateDescription (modellingRule, high)
- Iec61850DOAbbreviationLiteralsNeverUsedInDOName (modellingRule, high)
- Iec61850DOAttributesWithSameNameDifferentType (modellingRule, high)
- Iec61850ConditionLiteralsNeverUsedAsConstraints (modellingRule, high)

IEC61850 only

Available rules in OperationValidator (category, severity):
- OperationsWithUpperCaseName (namingRule, medium)
- OperationsWithUnallowedStereotype (modellingRule, high)
- OperationParametersWithUnallowedStereotype (modellingRule, high)
- OperationsWithInvalidReturnTypeNull (modellingRule, high)
- OperationsWithInvalidArgTypeNull (modellingRule, high)
- OperationsWithInvalidExcTypeNull (modellingRule, high)
- OperationsWithUnallowedTagNames (modellingRule, high)
- OperationParametersWithUnallowedTagNames (modellingRule, high)
- OperationsMissingDoc (documentationRule, medium)
- OperationParametersMissingDoc (documentationRule, medium)
- OperationsWithBadDocStart (documentationRule, low)
- OperationParametersWithBadDocStart (documentationRule, low)
- OperationsWithBadDocEnd (documentationRule, low)
- OperationParametersWithBadDocEnd (documentationRule, low)
- OperationsWithBadCharacterInName (namingRule, high)
- OperationParametersWithBadCharacterInName (namingRule, high)

Available rules in AssociationValidator (category, severity):
- AssociationsWithExplicitDirection (modellingRule, high)
- AssociationsWithRoleBadDirection (modellingRule, high)
- AssociationsWithDoc (documentationRule, low)
- AssociationsWithSameDocOnBothEnds (documentationRule, medium)
- AssociationsWithName (namingRule, medium)
- AssociationsWithUnallowedStereotype (modellingRule, high)
- AssociationEndsWithUnallowedStereotype (modellingRule, high)
- AssociationsMissingInformativeStereotype (modellingRule, high)
- AssociationsWithUnallowedTagNames (modellingRule, high)
- AssociationEndsWithUnallowedTagNames (modellingRule, high)
- AssociationsWithNoMultiplicity (modellingRule, high)
- AssociationsWithWrongSource (modellingRule, high)
- Iec61850AssociationsThatShouldBePrivate (modellingRule, high)
- Iec61850AssociationsWithDifferentEndVisibility (modellingRule, high)
- AssociationEndsMissingDoc (documentationRule, medium)
- AssociationEndsWithBadDocStart (documentationRule, low)
- AssociationEndsWithBadDocEnd (documentationRule, low)
- AssociationEndsWithBadCharacterInName (namingRule, high)
- CimAssociationEndsNameStartingWithLowerCase (namingRule, high)
- CimAssociationEndsNameShouldBePlural (namingRule, high)
- CimAssociationEndsNameShouldBeSingular (namingRule, high)

Available rules in DependencyValidator (category, severity):
- DependenciesWithUnallowedStereotype (modellingRule, high)
- DependenciesWithUnallowedDirection (modellingRule, high)
- DependenciesWithUnallowedTagNames (modellingRule, high)

Available rules in DiagramValidator (category, severity):
- DiagramsWithBadOrientation (formatting, low)
- DiagramsWithUnallowedStereotype (modellingRule, high)
- DiagramsMissingDoc (documentationRule, medium)
- DiagramsWithBadDocStart (documentationRule, low)
- DiagramsWithBadDocEnd (documentationRule, low)
- DiagramsWithBadCharacterInName (namingRule, high)

Validation:

Console logging

Extract from validation log with base-small.eap UML model

- **ERROR Found 3 CIM compound classes** with <diagnosis>
- Next **3** lines identify **classes** with **error**/warning and details of error/warning

```
[main] INFO  ====== Validating 93 (of 93) classes:
[main] WARN  Found 2 classes with unexpected embedded elements; they are present in the model repository, but not kept in the in-memory model:
[main] WARN    root class Informative::InfClassContainingEmbeddedClass: [(1496) WG13 CIM INF other InfClassContainingEmbeddedClass.EmbeddedClass]
[main] WARN    class Core::PowerSystemResource: [(1444) WG13 CIM state PowerSystemResource.DummyState]
[main] ERROR Found 1 enumeration classes with no literals. Attributes with that type can only be null:
[main] ERROR   (1480) WG14 CIM enumeration <<enumeration>> Other::EmptyEnum used by: []
[main] ERROR Found 3 CIM compound classes with no attributes. Attributes with that type can only be null:
[main] ERROR   (1532) OTHER_CIM CIM INF compound <<Compound>> Informative::SomeSimpleType used by: [OTHER_CIM Ext1::Pear.typeIsInformative]
[main] ERROR   (1538) WG14 CIM compound <<Compound>> Other::EmptyCompound used by: [WG14 Other::AnotherBadDatatype.multiplier]
[main] ERROR   (850) WG13 CIM compound <<Compound>> Core::OperatingParticipant used by: []
[main] WARN  Found 1 enumeration classes with single literal. Does it make sense to keep the enumerated type with a single literal?:
[main] WARN    (1533) WG13 CIM enumeration <<enumeration>> Domain::WithSingleLiteral used by: []
```

Also available a dedicated .csv report:

- Open it with a spreadsheet app for easier sorting, filtering and analysis

- But, if we keep our std CIM and IEC 61850 UML clean, this may not be necessary!

Validation:

Fine tuning

If you leave **scope** property empty, the full content of the UML will be validated

- To validate only some top level packages (per IEC WG), specify them in a comma-separated list
  - Example is for validating IEC61970 and IEC61968, everything else does not get validated
- **Recommendation: Before releasing std model, do full validation, to ensure nothing has been broken**

Sub-options for validation, to skip validation for one or more type of UML element

- By default, nothing is skipped
- To skip validating something, set it to true:
  - Example for validating only associations

Sub-option for validation, to skip individual rules

- ... but skip rules reporting associations having doc, and association ends missing doc

```
model.filename       = base-small.eap

validation.on        = true
validation.scope     = WG13, WG14

validation.packages.off       = true
validation.classes.off        = true
validation.associations.off   =
validation.attributes.off     = true
validation.operations.off     = true
validation.dependencies.off = true
validation.diagrams.off       = true

validation.rules.off =                  \
    AssociationsWithDoc                 \
    AssociationEndsMissingDoc
```

Features: UML model statistics

• • •

Numbers and more.

Statistics:

Minimum configuration

You must specify the UML model file name and enable statistics.

Copy your own model file(s) into the project's **input** directory.

```
model.filename   = base-small.eap

statistics.on        = true
```

Statistics:

Overview

Currently, these kinds of statistics get logged to the console:

- Counts of UML constructs: classes, packages, diagrams, tags, etc.
- On CIM-specific constructs for classes and attributes (e.g., CIM data types, compounds, association ends, …)
- On IEC 61850-specific constructs for classes and attributes (e.g., LNs, CDCs, packed lists, operations, …), underlying modelling (e.g., classes and attributes with constraints, etc.), and since 10v10: DO name decomposition and inverse (usage of abbreviations by DOs)
- Tag names and where they are used
- Items with constraints
- Identified UML version classes and name spaces
- On actual (direct and derived) dependencies among packages

Statistics:

On CIM-specific constructs

Example of CIM-specific constructs from base-small.eap:

- Total number of elements in the whole model
- Total number of elements per top-level package (per WG)
- Counts also informative elements

```
[main] INFO  ====== Stats per nature and per owner for 72 packages (of 72):
[main] INFO  ------ CIM statistics:
[main] INFO  [WG13]
[main] INFO  5 packages (72/72) - 1 informative:
[main] INFO    1 top package (per WG)
[main] INFO    4 sub-package (any below top)
[main] INFO  44 classes (356/356) - 2 informative:
[main] INFO    4 primitive class
[main] INFO    9 enumeration class
[main] INFO    8 CIM datatype class
[main] INFO    1 compound class
[main] INFO    5 root class
[main] INFO    17 class
[main] INFO  144 attributes (689/689):
[main] INFO    28 primitive attribute
[main] INFO    6 CIM datatype attribute
[main] INFO    90 enumeration literal
[main] INFO    20 other attribute
[main] INFO  21 associations (95/95):
[main] INFO    9 aggregation
[main] INFO    12 simple association
[main] INFO  5 operations (14/14):
[main] INFO    1 void op()
[main] INFO    1 T[] op()
[main] INFO    3 T op()
[main] INFO  3 dependencies (72/72):
[main] INFO    3 interPackage dependency
[main] INFO  9 diagrams (103/103) - 1 informative:
[main] INFO    8 class diagram
[main] INFO    1 statechart diagram
[main] INFO  12 tag names (19/19):
[main] INFO    3 dummyCimTag
[main] INFO    1 tag
[main] INFO    1 assocTag
[main] INFO    1 srcTag
[main] INFO    1 CE-TermAssoc
[main] INFO    1 MoreAssoc
[main] INFO    1 Role1
[main] INFO    1 AnotherRole2
[main] INFO    1 targetEndTag
[main] INFO    1 Role2
[main] INFO    2 throws
[main] INFO    2 someTag
```

Statistics:

On IEC 61850-specific constructs

Example of IEC 61850-specific constructs from base-small.eap:

- Total number of elements in the whole model
- Total number of elements per top-level package (per WG)

```
[main] INFO  ------ IEC61850 statistics:
[main] INFO  [WG10]
[main] INFO  50 packages (72/72) - 2 informati
[main] INFO    1 top package (per WG)
[main] INFO    49 sub-package (any below top)
[main] INFO  286 classes (356/356):
[main] INFO    14 basic class
[main] INFO    4 packed class
[main] INFO    13 enumeration class
[main] INFO    5 coded enumeration class
[main] INFO    2 abbreviation enumeration clas
[main] INFO    1 presence condition enumeratio
[main] INFO    3 coded enumeration DA class
[main] INFO    8 enumeration DA class
[main] INFO    4 packed list DA class
[main] INFO    10 primitive DA class
[main] INFO    5 composed DA class
[main] INFO    4 coded enumeration FCDA class
[main] INFO    15 enumeration FCDA class
[main] INFO    53 FCDA class
[main] INFO    7 enumeration CDC class (derive
[main] INFO    20 primitive CDC class
[main] INFO    4 composed CDC class
[main] INFO    17 LN class
[main] INFO    11 function (61850-5) class
[main] INFO    83 other 61850 class
[main] INFO    3 unknown 61850 class
[main] INFO  526 attributes (689/689):
[main] INFO    52 basic attribute
[main] INFO    4 packed attribute
[main] INFO    21 coded enumeration literal
[main] INFO    7 abbreviation enumeration literal
[main] INFO    30 presence condition enumeration lit
[main] INFO    3 coded enumeration DA attribute
[main] INFO    9 enumeration DA attribute
[main] INFO    4 packed list DA class
[main] INFO    23 attribute on any DA whose type is
[main] INFO    14 attribute on composed DA whose typ
[main] INFO    2 coded enumeration FCDA attribute
[main] INFO    5 enumeration FCDA attribute
[main] INFO    23 FCDA attribute
[main] INFO    8 enumerated DO (derived)
[main] INFO    10 data object (attribute on LN)
[main] INFO    1 sub-data object (attribute on compo
[main] INFO    171 enumeration literal
[main] INFO    139 other attribute
[main] INFO  60 associations (95/95):
[main] INFO    18 composition
[main] INFO    42 simple association
[main] INFO  8 operations (14/14):
[main] INFO    1 T[] op()
[main] INFO    7 T op()
[main] INFO  68 dependencies (72/72):
[main] INFO    48 interPackage dependency
[main] INFO    20 interClass dependency
[main] INFO  79 diagrams (103/103) - 3 informative:
[main] INFO    7 custom diagram
[main] INFO    63 class diagram
[main] INFO    3 statechart diagram
[main] INFO    6 package diagram
[main] INFO  6 tag names (19/19):
[main] INFO    6 scl
[main] INFO    11 iecRef
[main] INFO    11 ieeeRef
[main] INFO    11 rsName
[main] INFO    2 scl:emptyValue
[main] INFO    6 moveAfter
```

Statistics:

On actual dependencies

On actual (**direct and derived**) dependencies among top-level packages (i.e., WG owners), through:

- Inheritance (e.g., from PowerSystemResource)
- Type of attribute (e.g., usage of datatypes, enums, primitives, compounds)
- Associations
- Hand-drawn package and class dependencies
- Class' operation parameters and exceptions (UML of IEC61850 only)

(CIM only) Inheritance from IdentifiedObject and dependency on stereotyped types from Domain package not shown on purpose:

- To show them as well, set the two ignore* properties to false, or leave them empty

```
[main] INFO  ====== Cross-package stats for 32 packages (of 32):
[main] INFO  Cross-owner links:
[main] INFO    1 class inheritance:
[main] INFO    (excluded inheritance from IdentifiedObject)
[main] INFO      [WG13 Core::BaseVoltage, WG14 Other::BadDatatype]
[main] INFO    4 class's attribute types:
[main] INFO    (excluded types from Domain package)
[main] INFO      [OTHER_CIM Informative::HasIllegalTypeForAttr, WG13 Core::Bay]
[main] INFO      [WG13 Domain::ActivePowerChangeRate, OTHER_CIM NullCIM::NullCIM]
[main] INFO      [WG14 Other::MyClass, OTHER_CIM NullCIM::NullCIM]
[main] INFO      [WG14 Other::AttrDuplication, OTHER_CIM NullCIM::NullCIM]
[main] INFO    1 class's operation parameters and exceptions:
[main] INFO      [WG13 Core::PowerSystemResource, OTHER_CIM NullCIM::NullCIM]
[main] INFO    0 class-to-class (hand-drawn) dependency:
[main] INFO    4 class-to-class associations:
[main] INFO      [WG14 Other::MyClass, OTHER_CIM Informative::Class1]
[main] INFO      [WG10 SCL::FictSCLClass, WG14 Other::MyClass]
[main] INFO      [OTHER_IEC61850 Ext2::Animal, OTHER_CIM Ext1::Fruit]
[main] INFO      [OTHER_IEC61850 Ext2::Animal, OTHER_CIM Ext1::Fruit]
[main] INFO    1 package-to-package (hand-drawn) dependency:
[main] INFO      [WG13 IEC61970::Topology, WG14 IEC61968::Other]
```

statistics.cim.**ignoreIdObjectInheritance**	= true
statistics.cim.**ignoreDomainClassAttributes**	= true

Statistics:

TODOs

Configuration options to disable some of the output:

- sometimes it's overwhelming, but may be also very useful

Better presentation and potentially storage:

- In-memory representation of the UML from .eap file contains a lot of analysed information about the model
- It all gets logged with DEBUG level (i.e., stored in the log file at each run) – see examples below for ConnectivityNode, ApparentPower, BasicIntervalSchedule

```
2010-06-06 14:05:21,984 [main] DEBUG UmlModel - (796) WG13 CIM class Topology::ConnectivityNode, 1
superclasses=[IdentifiedObject], 4 associations; associated classes (bi-directional): asTarget=[Topology::BusNameMarker,
Core::Terminal] asSource=[Core::ConnectivityNodeContainer, Topology::TopologicalNode]
…
2010-06-06 14:05:21,984 [main] DEBUG UmlModel - (616) WG13 CIM Datatype <<Datatype>> Domain::ApparentPower, 3
attributes; afferent classes: byAttr=[BasePower];; efferent classes: byAttr=[Float, UnitMultiplier, UnitSymbol]
…
2010-06-06 14:05:21,984 [main] DEBUG UmlModel - (865) WG13 CIM class Core::BasicIntervalSchedule, 1
superclasses=[IdentifiedObject], 2 subclasses=[IrregularIntervalSchedule, RegularIntervalSchedule], 5 attributes; efferent classes:
byAttr=[AbsoluteDateTime, UnitMultiplier, UnitSymbol]
```

afferent = depends on me
efferent = I depend on

Features:
MS Word doc generation from UML (and from CIM XSD profiles*)

• • •

AKA : A very, very big pain…

*being implemented

MS Word doc generation from UML:

Minimum configuration*

You must specify the UML model file name and enable doc generation

- Ensure profiles.docgen.on is ***not*** set to true (= leave it empty or set it to false)
- (until we have new implementation) Ensure docgen.word.useDocFormat = true

You must specify also the input (template) and the output (result) MS Word file names.

Copy your own model file(s) and template(s) into the project's **input** directory.

```
model.filename                = base-small.eap
model.builder                 = sqlxml
docgen.on                     = true
profiles.docgen.on            =

docgen.word.inTemplate        = base-small-template.doc
docgen.word.outDocument       = base-small.doc
docgen.word.saveReopenEvery   = 12
docgen.word.useDocFormat      = true
** docgen.word.analysePlaceholders =
*** docgen.word.useHyperlinks =
```


See slide "MS Word speed considerations"

See slide "File and package placeholders"

See slides "Hyperlinks" and "MS Word speed considerations"

* UML of IEC61850 needs more than this, see config61850.properties and doc in Configuration class

** Started (but not finished) implementing faster MS Word docgen, that will be default

*** Since v02v01

MS Word doc generation from UML:

Overview

Template file is a regular MS Word document (**not** Word .dot template):

- **You** put in that template jCleanCim-recognised placeholders (see next slide)
- They control what to pick from the UML model and print into MS Word document

When generating documentation, jCleanCim will:

- Copy your template file into the projects **output** directory, created automatically the first time you run the document generation,
- Rename the copied file as given in the properties file, and
- Fill it with the contents from the EA model in place of placeholders found.

You can safely run document generation several times with the same name of the output file, without overwriting existing output files:

- If the output file exists, jCleanCim will rename it by appending a unique identifier
- The disadvantage is that you will need to delete those discarded files from the output directory from time to time, but at least nothing gets lost without your control

Resulting file is available in the project's **output** directory.

MS Word doc generation from UML: Placeholders

The tokens enclosed in curly braces are the names of UML elements designating what needs to be inserted in place of the whole placeholder.

- Ensure there are no spaces around the dots '.' (that MS Word "smartly" introduces for you on copy/paste)
- Placeholders assume that the names (of packages, classes and diagrams-within-package) are unique within the model; otherwise, the first item is picked and that may not be what you want…

startUmlDiagram.{packageName}.{diagramName}.endUml
* startUmlDiagNote{packageName}.{diagramName}.endUml
startUmlAttribute.{className}.{attributeName}.endUml
startUmlFile..endUml
startUmlPackage.{packageName}.endUml
* startUmlClass.{packageName.className}.endUml

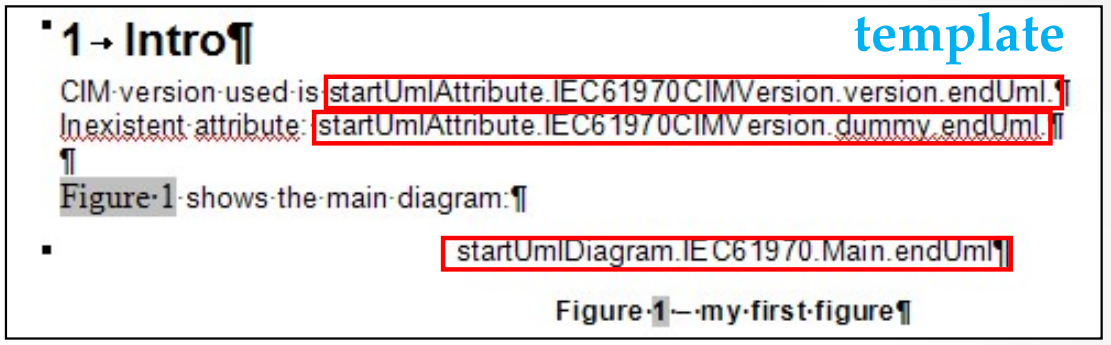
* startUmlstartUmlIec61850NsName.{className}.endUml	(IEC 61850-7-*, for name space name)
startUmlPresenceConditions.{packageName}.endUml	(IEC 61850-7-2, 7-3, 7-4xx, for presence conditions)
startUmlFCs.{packageName}.endUml	(IEC 61850-7-2, 7-3, for FC table)
startUmlTrgOps.{packageName}.endUml	(IEC 61850-7-2, for TrgOp table)
startUmlSclEnums.{packageName}.endUml	(IEC 61850-7-2, 7-3, 7-4xx, for SCL enums)
startUmlAbbreviations.{packageName}.endUml	(IEC 61850-7-4xx, for DO abbreviations)
startUmlDataIndex.{packageName}.endUml	(IEC 61850-7-4xx, for data semantics tables)
startUmlLNMapPackage.{packageName}.endUml	(IEC 61850-7-4, for Domain61850 tables)

* since 02v01

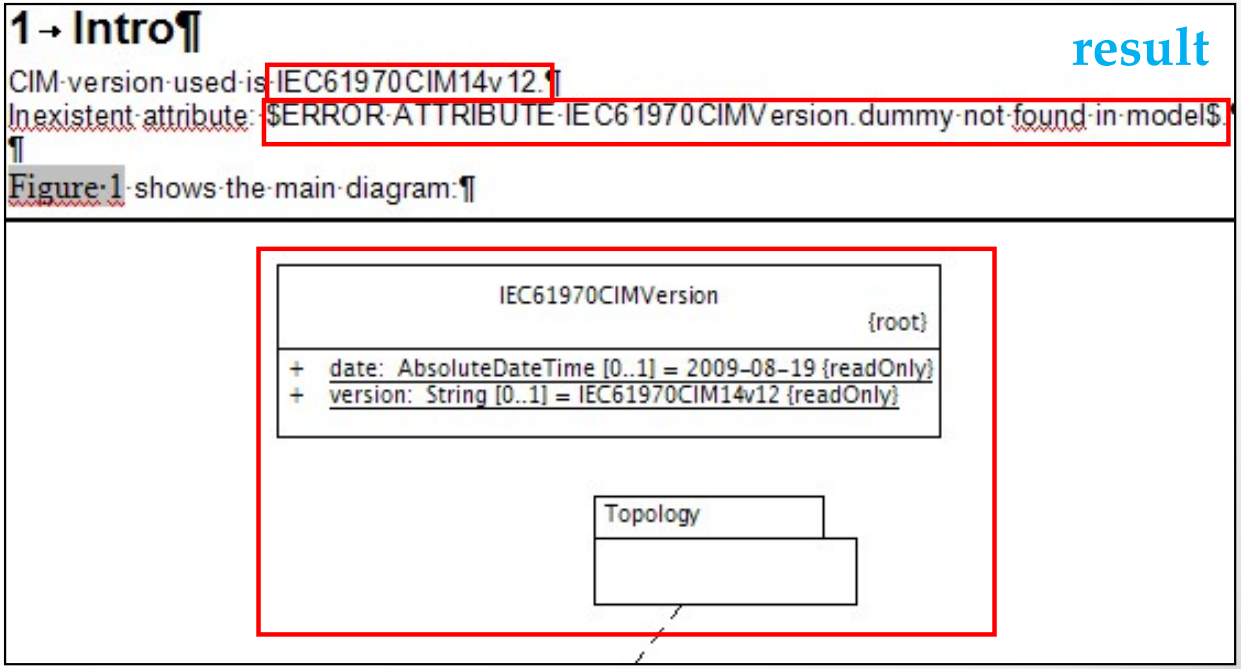
MS Word doc generation from UML:

Attribute and diagram placeholders

Placeholder from template gets replaced with some content



If placeholder is not recognised or has typos, error gets printed instead of content



MS Word doc generation from UML:

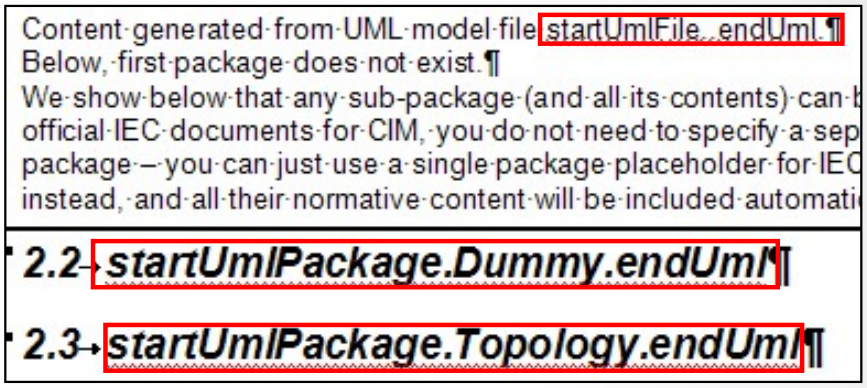
File and package placeholders

If problem with the placeholder for package (in the heading paragraph), only error gets printed instead of the package content

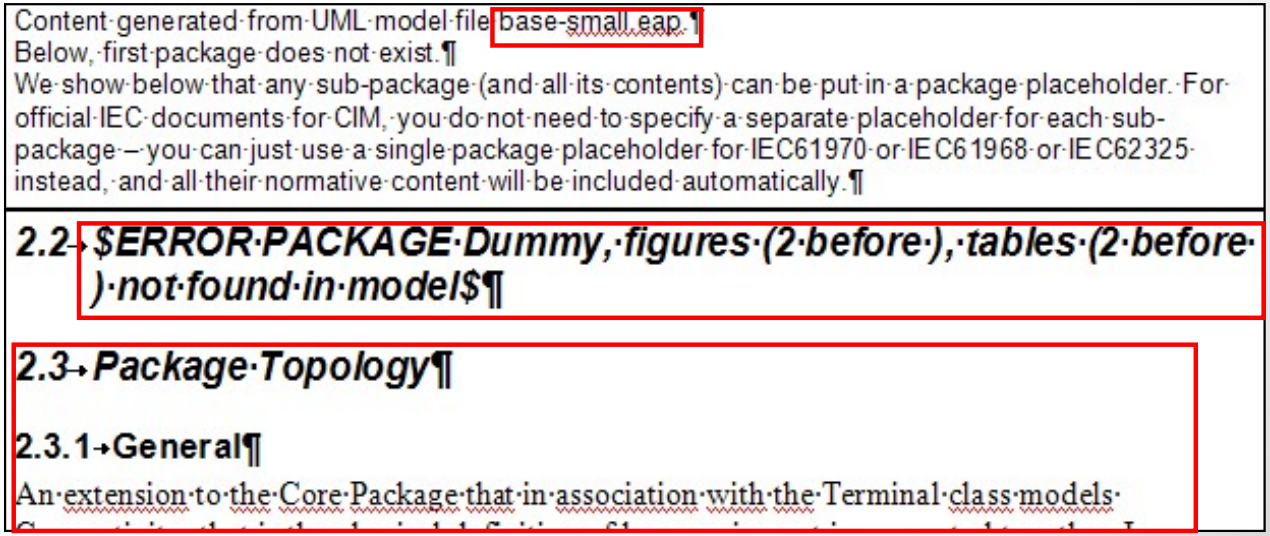
Doc generation (in particular for UML package content, recursively) takes long:

- **In template debugging phase,** to ensure you have correct placeholders for packages, set property **docgen.word.analysePlaceholders =true**; This will skip (time consuming) printing of the full content of packages
- **After debugging the template**, leave docgen.word.**analysePlaceholders empty to get the full content printed!**

template



result



MS Word doc generation from UML:

Diag. note and explicit class placeholders

Available since 02v01.

Diagram description normally gets printed automatically after a diagram, from within the containing package or class.

DiagNote placeholder inserts the description of the diagram :

- Useful for some extension models, if you want e.g. to have a two-column table with a diagram in one column and its description in another.

Classes normally get printed automatically from within a package. Sometimes you may want to selectively print one or more classes.

Class placeholder inserts explicitly the class content:

- Ensure to put this placeholder in a heading to get proper indentation.

template

Note for existing diagram comes next:
startUmlDiagNote.IEC61970.Main.endUml

Note for inexisting diagram comes next:
startUmlDiagNote.IEC619_70.Main.endUml

One existing CIM class, one existing 61850 class, one enumeration a

6.1 startUmlClass.Core.BasePower.endUml

result

Note for existing diagram comes next:
'This diagram shows all 61970 packages and their logical dependencies.
Test bold ignored.

Note for inexisting diagram comes next:
$ERROR DIAG_NOTE IEC619_70, Main not found in model$

One existing CIM class, one existing 61850 class, one enumeration and one inexisting

6.1 BasePower

the BasePower class defines the base power used in the per unit calculations. Table 111 shows all attributes of BasePower.

Table 111 – Attributes of Core::BasePower

name	mult	type	description
basePower	0..1	ApparentPower	definition of base power.
aliasName	0..1	String	inherited from: IdentifiedObject
mRID	0..1	String	inherited from: IdentifiedObject

MS Word doc generation from UML:

IEC 61850 name space placeholder

Available since 02v01.

IEC 61850 has specific format how to create the so-called name space name from attributes in the name-space UML class.

IEC61850NsName placeholder inserts that normative string properly formatted.

(Note: work is in progress for CIM canonical model and profile namespaces)

template

Namespace names:
IEC61850 namespace name: "startUmlIec61850NsName.IEC61850_7_2Namespace.endUml"
IEC61850 namespace name: "startUmlIec61850NsName.IEC61850_7_3Namespace.endUml"
IEC61850 namespace name: "startUmlIec61850NsName.IEC61850_7_4Namespace.endUml"
IEC61850 namespace name:
"startUmlIec61850NsName.IEC61850_7_420Namespace.endUml"

result

Namespace names:
IEC61850 namespace name: "IEC61850-7-2:2007"
IEC61850 namespace name: "IEC61850-7-3:2011B"
IEC61850 namespace name: "IEC61850-7-4:2009A"
IEC61850 namespace name: "$ERROR IEC61850_NSNAME IEC61850_7_420Namespace not found in model$"

MS Word doc generation from UML:

Data index placeholder

Required for IEC61850 documents

- Can be handy for CIM series as well, and for custom extensions
- Can be used for "model debugging" and consistency check

Prints all usages of a name for attribute

- From base-small.eap example: attribute **basePower** used in classes **BaseVoltage** and **BasePower**

Currently, only attributes

- If needed, (named) association ends could be added

For this to work, you must specify the desired package name (or multiple comma-separated package names) in the property validation.**packagesWithDataIndex**

- For this base-small.eap example, we have set validation.packagesWithDataIndex = Core
- To print this index for base CIM, set validation.packagesWithDataIndex = IEC61970

template

3 Optional data index (e.g., in appendix)

3.1 *startUmlDataIndex.Core.endUml*

result

4 Optional data index (e.g., in appendix)

This has been implemented for IEC61850-7-3 and -7-4, but can also be used for assessing CIM attributes and making their documentation and types uniform. Currently, we print only attributes; if needed, we could do the same for association ends.

4.1 Data semantics

Table 78 shows all attributes defined on classes of Core package.

Table 78 – Attributes defined on classes of Core package

Name	Type	(Used in) Description
aliasName	String	(IdentifiedObject) The aliasName is free text human readable name of the object alternative to IdentifiedObject.name. It may be non unique and may not correlate to a naming hierarchy.
basePower	Boolean, ApparentPower	(BaseVoltage) This is to test whether we print correctly multiple attributes of the same name (defined on different classes) within the data index table. (BasePower) definition of base power.
bayEnergyMeasFlag	Boolean	(Bay) Indicates the presence/absence of energy measurements.

MS Word doc generation from UML:

Automatically added items (1/2)

Sub-clause "General"

- To avoid hanging paragraphs (i.e., text without containing clause)

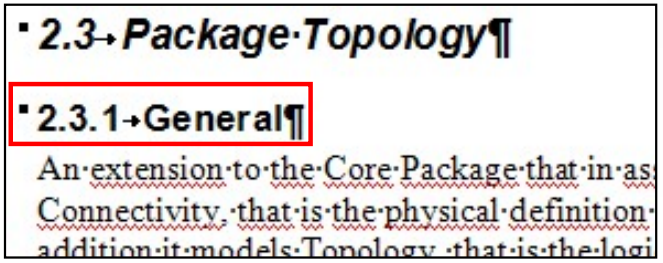
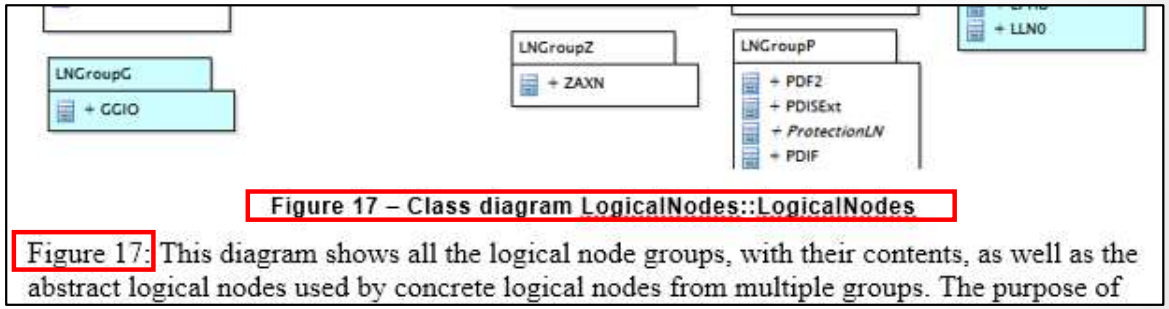


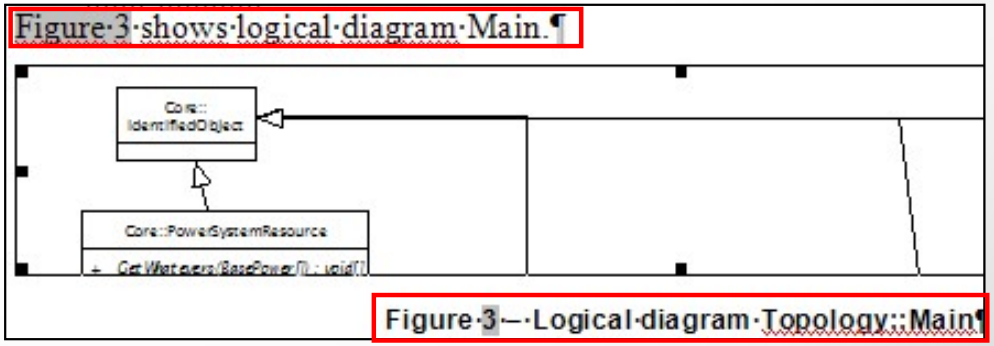
Figure reference and text, and figure caption

- Automatically numbered, consistent with existing figures in the template
- Legacy look, if desired, needs to be enabled: docgen.word.introToFigureBefore = true

New look (default) :



Legacy look:



MS Word doc generation from UML:

Automatically added items (2/2)

Table reference and text for attributes and association ends

- Automatically numbered, consistent with existing figures in the template

Table 59 shows all attributes of TopologicalIsland.

Table 59 – Attributes of Topology::TopologicalIsland

name	mult	type	description
constrained5	0..1	Integer	
nonConstrained	0..1	Integer	
mRID	0..1	String	inherited from: IdentifiedObject
name	0..1	String	inherited from: IdentifiedObject
localName	0..1	String	inherited from: IdentifiedObject
pathName	0..1	String	inherited from: IdentifiedObject
aliasName	0..1	String	inherited from: IdentifiedObject
description	0..1	String	inherited from: IdentifiedObject

Table 60 shows all association ends of TopologicalIsland with other classes.

Table 60 – Association ends of Topology::TopologicalIsland with other classes

mult from	name	mult to	type	description
0..1	AngleRef_TopologicalNode	0..1	TopologicalNode	The angle reference for the island. Normally there is one TopologicalNode that is selected as the angle reference for each island. Other reference schemes exist, so the association is optional.
1..1	TopologicalNodes	?..?	TopologicalNode	A topological node belongs to a topological island

MS Word doc generation from UML:

Fancy IEC 61850 table formats

Some IEC 61850 tables want special formatting

- With table title
- Sub-headings
- With special reference and element counts for array types
- With special presence conditions (instead of UML multiplicities)
- Etc.

Table 37 – Attributes of HDEL

cdcId = HDEL, UML class name = HDEL					
Attribute name	**Attribute type**	**FC**	**TrgOp**	**(Value/Value range) Description**	**PresCond**
SubDataObject					
phsABHar	ARRAY 0...maxPts-1 OF CMV			Array of harmonic and subharmonics, or interharmonic values related to phase A to phase B.	M
phsBCHar	ARRAY 0...maxPts-1 OF CMV			Array of harmonic and subharmonics, or interharmonic values related to phase B to phase C.	O
phsCAHar	ARRAY 0...maxPts-1 OF CMV			Array of harmonic and subharmonics, or interharmonic values related to phase C to phase A.	O
DataAttribute for configuration, description and extension					
numHar	INT16U	CF	dchg	inherited from: HarmonicMeasurandCDC	M
numCyc	INT16U	CF	dchg	inherited from: HarmonicMeasurandCDC	M
evalTm	INT16U	CF	dchg	inherited from: HarmonicMeasurandCDC	M
angRef	PhaseAngleReferenceKind	CF	dchg	Angle reference, indicating the quantity that is used as reference for the respective phase angle (e.g., 'phsABHar[i].ang'), or that the values are	O

MS Word doc generation from UML: Hyperlinks

Available since 02v01.

From IEC 61850 community push:

- To enable this feature, set docgen.word. useHyperlinks = true
- Default is false, because it is very time consuming (due to a second pass, to replace bookmarks with hyperlinks)!
- Hyperlinks are generated to refer to any class (and some special enumeration literals) that are defined within one MS Word document
  - E.g. CtlModeKind, BasePrimitiveCDC, MFsbo
- For classes not printed in the given MS Word document, there are no hyperlinks
  - E.g. INT32U, VisString255

del	CtlModelKind	CF	dchg	See 'SPC.ctlModel'.	M
meout	INT32U	CF	dchg	See 'SPC.sboTimeout'.	MFsbo
ass	SboClassKind	CF	dchg	(default=operate-once) See 'SPC.sboClass'.	O
l	INT8	CF	dchg	Minimum setting for 'valWTr.posVal' below which 'ctlVal'='lower' will have no effect.	O
al	INT8	CF	dchg	Maximum setting for 'valWTr.posVal' above which 'ctlVal'='higher' will have no effect.	O
imeout	INT32U	CF	dchg	See 'SPC.operTimeout'.	MFenhanced
	VisString255	DC		inherited from: BasePrimitiveCDC	O
	Unicode255	DC		inherited from: BasePrimitiveCDC	O
me	VisString255	EX		inherited from: BasePrimitiveCDC	O

MS Word doc generation from UML:

MS Word styles considerations

IEC templates contain IEC-specific styles.

For custom extensions, you need not use these:

- jCleanCim tries to use IEC styles, and if those are not defined, it uses default MS Word styles – see next slide

Essential:

- **Use correct styles for paragraphs with *figure and table captions* in the template**
- jCleanCim must deduce the number of figures and tables already existing in the template to calculate on the fly the correct numbering for new figures and tables (when inserting/appending the documentation for the UML model elements and diagrams)
- If jCleanCim throws an exception during document generation, it is very likely that the MS Word threw exception due to wrong / inexistent / negative number for the figure or table caption
- Note: We cannot check those numbers from within the code, because the MS Word automation API does not provide reliable access to them. In the worst case, when we catch an exception from MS Word, we attempt to gracefully exit, after closing both the MS Word document and the EA model file.

MS Word doc generation from UML:

IEC styles mappings to MS Word defaults

Extract from the code:

IEC styles	MS Word defaults
para("PARAGRAPH",	"Normal"),
fig("Picture",	"Normal"),
figcapt("FIGURE-title",	"Caption"),
tabcapt("TABLE-title",	"Caption"),
tabhead("TABLE-col-heading",	"Normal"),
tabcell("TABLE-cell",	"Normal"),
h1("Heading 1",	"Heading 1"),
h2("Heading 2",	"Heading 2"),
h3("Heading 3",	"Heading 3"),
h4("Heading 4",	"Heading 4"),
h5("Heading 5",	"Heading 5"),
h6("Heading 6",	"Heading 6"),
h7("Heading 7",	"Heading 7"),
h8("Heading 8",	"Heading 8"),
h9("Heading 9",	"Heading 9");

MS Word doc generation from UML:

MS Word speed considerations (1/2)

MS Word documentation takes very long for big models !

- Profiling of jCleanCim-01v04 and then jCleanCim-01v05 has shown that it is **not** due to Java code or Java-COM bridge
- It is certain MS Word operations that are slow
- In particular, inserting captions for figures and tables, takes exponential time as the number of figures and tables grows
  - It is ***impossible*** to disable MS Word doing that numbering, while preserving ability to print tables of figures/tables

Since jCleanCim-01v05, we significantly improved the speed

- From time to time, we close the output document, and reopen it
- This seems to reset the MS Word's "numbering memory"

Since jCleanCim-01v08, some more improvement:

- Only if your template has been saved as Office 2007/2010/2013 document (.docx), **without** compatibility options
- Programmatic disabling of field update seems to work only then

MS Word doc generation from UML:

MS Word speed considerations (2/2)

- **Use the** docgen.word.saveReopenEvery **configuration option**
  - Supplied config.properties file contains the "magic" numbers for tested documents, as for our development environment
  - Try to slightly increase/decrease the value to see whether there is any improvement in your environment and for your document size

- Enable hyperlink creation only for the very end of your development cycle (i.e., just before generating the final document), as it is very time consuming

- If possible, save your template as Office 2007/2010/2013 document (.docx), **without** compatibility options

- Disable change tracking in your template

MS Word doc generation from UML:

Misc

For IEC61850 document generation, more properties need to be set:

- See config/config61850.properties file and javadoc of the Configuration class

jCleanCim doc generation design:

- The UML packages content to be printed in MS Word actually gets calculated and stored in in-memory objects **before** interacting with MS Word, on purpose:
  - This processing is very fast (sub-second)
  - It is also detached from the .eap file
  - We now collect also content for creating XML documents

Features:
XML doc generation from UML

• • •

This feature was driven by IEC61850 Web Publishing efforts
(and works for CIM as well)

XML doc generation from UML:

Minimum configuration*

You must specify the UML model file name and enable doc generation:

- Ensure profiles.docgen.on is ***not*** set to true (make it false or leave empty)

You must specify also the two output (result) XML file names.

Copy your own model file(s) into the project's **input** directory:

- IECDomain.xsd schema is already available in the project's **input** directory

If you want to limit the scope (generate only some name spaces), use the scope variable (similar as for validation).

```
model.filename          = base-small.eap
model.builder           = sqlxml

docgen.on               = true
profiles.docgen.on      = false

docgen.xml.scope        = WG10, WG13
docgen.xml.outSpec      = base-small-tool01v08-spec.xml
docgen.xml.outDoc       = base-small-tool01v08-doc.xml
```

Empty value generates everything found in the UML model.
This example generates only WG10 and WG13 name spaces

* UML of IEC61850 needs more than this, see config61850.properties and doc in Configuration class

XML doc generation from UML:

Overview

When generating documentation, jCleanCim will:

- Copy the schema file to the **output** directory, created automatically the first time you run the document generation,
- Create .xml files as given in the properties file,
- Fill them with the contents from the EA model, and
- Save all the diagrams in the **output/pics** directory.

You can safely run document generation several times with the same name of the output files, without overwriting existing output files:

- If the output files (or the schema file) exist in the output directory, jCleanCim will rename them by appending a unique identifier
- The disadvantage is that you will need to delete those discarded files from the output directory from time to time, but at least nothing gets lost without your control

Resulting files are available in the project's **output** directory.

For a release, zip the two produced instance files, the schema and the pics directory.

XML doc generation from UML:

Specification and documentation XML

By design, we generate two separate instance files:

- Documentation, with translatable (potentially formatted) strings with an identifier
- Specification, with normative, non-translatable content, with references to documentation identifiers

One can have any number of translated documentation files (with correct identifiers) that can be combined with the specification content

```
<iec:IECDomainDoc xmlns:iec="http://iec.ch/TC57/UML/2012/IECDomain" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
http://iec.ch/TC57/UML/2012/IECDomain IECDomain.xsd">
 <iec:IEC61850Domain>
  <iec:Doc id="daStatusLabel"><![CDATA[status]]></iec:Doc>
  <iec:Doc id="daMeasLabel"><![CDATA[measured attributes]]></iec:Doc>
  <iec:Doc id="daCtlMirrorLabel"><![CDATA[control mirror]]></iec:Doc>
  <iec:Doc id="daSubstLabel"><![CDATA[substitution and blocked]]></iec:Doc>
```

```
<iec:IECDomainSpec xmlns:iec="http://iec.ch/TC57/UML/2012/IECDomain" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
http://iec.ch/TC57/UML/2012/IECDomain IECDomain.xsd">
 <iec:IEC61850Domain>
  <iec:PrettyStrings>
  <iec:NameSpace id="IEC61850-7-2" version="2007" umlVersion="">
   <iec:FunctionalConstraints introductionID="IEC61850_7_2::FunctionalConstraints.90.fc.introduction" captionID="IEC61850_7_2::FunctionalConstraints.90.fc.caption" titleID="
IEC61850_7_2::FunctionalConstraints.90.fc.title" name="FunctionalConstraints" aliasID="IEC61850_7_2::FunctionalConstraints.90.fc.alias" descID="
IEC61850_7_2::FunctionalConstraints.90.fc.desc">
    <iec:FC name="ST" aliasID="FunctionalConstraints::FcKind.ST.3356.alias" descID="FunctionalConstraints::FcKind.ST.3356.desc"/>
    <iec:FC name="MX" aliasID="FunctionalConstraints::FcKind.MX.3357.alias" descID="FunctionalConstraints::FcKind.MX.3357.desc"/>
```

XML doc generation from UML:

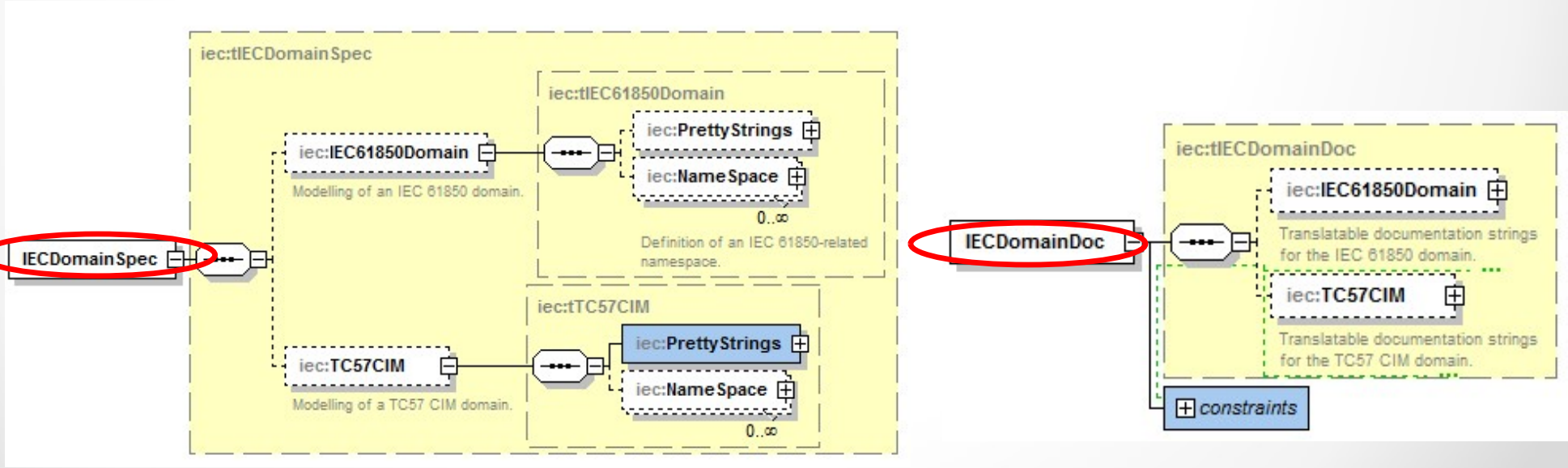
IECDomain.xsd

By design, we have a single schema with two elements defined:

- One for specification XML file, and one for documentation XML file

By design, the schema supports mix of CIM and IEC 61850 name spaces

- Each family under a separate element (for both specification and documentation files)
- You can select what you want to generate



XML doc generation from UML:

Status

This schema has been driven by IEC 61850 efforts, with objectives:

- to feed the IEC web-based publication, with the target, among many, to offer a way to download an XML machine readable file, representing the IEC 61850 data model definitions.
- to feed automatically a tool which would be able to semi-automatically check the compliance of some SCL configuration files based on the IEC 61850 data model.

jCleanCim tries to support both CIM and IEC 61850 model management needs, so:

- Whenever applicable, a feature required by one family of standards gets implemented for the other family of standards
- If we one day finally come to some kind of harmonisation at the UML level, we are ready ☺

Feedback on the schema design for the CIM community is welcome and asked for:

- **If there are experts in XSLT who would volunteer to provide stylesheets, that would be just great !**

Features:
CIM profiles vs. UML model cross-check*

• • •

*being implemented

Performance

• • •

Time considerations for reading from EA file and MS Word document generation

Each feature that has been run ends with time elapsed logged to the console:

```
[main] INFO  time=[0:00:23.415] built model from 'TC57CIM (CIM), IEC618
```

EA and MS Word automation API implementations are terribly slow:

- Java processing for validation, stats, doc collection and XML doc generation is of order of milliseconds to seconds
- The "party breakers'" times are given on the next slide

Times measured on ThinkPad T410 with Windows 7 64-bit, Java 7 32-bit, Office 2010, EA 9.3:

Processor:	Intel(R) Core(TM) i5 CPU	M 520 @ 2.40GHz 2.40 GHz
Installed memory (RAM):	8.00 GB (7.80 GB usable)	
System type:	64-bit Operating System	

So, plan your coffee and lunch breaks ☺

EA-dependent operations:

Execution times with jCleanCim-01v08

Improvements through new implementation enabled with option model.builder = sqlxml | db

- Full CIM model (iec61970cim16v17_iec61968cim12v06_iec62325cim02v07.eap):
  ~1520 classes, ~7680 attributes, ~1050 associations, ~90 dependencies, ~570 diagrams.
- Full IEC61850 model (wg10uml02v12-wg18uml02v10c-wg17uml02v09a-jwg25uml02v02a.eap):
  ~1670 classes, ~6250 attributes, ~85 associations, ~260 operations, ~380 dependencies, ~230 diagrams; plus tonnes of class and attribute constraints, tagged values, and markup in the documentation of elements.
- Small test model (base-small.eap):
  ~360 classes, ~700 attributes, ~95 associations, ~15 operations, ~70 dependencies, ~95 diagrams.

EA-dependent operation	**01v07** **useSql=false / useSql=true / -**	**01v08** **japi / sqlxml / db**
Open .eap file of any size (SSD hard disk)	5-10 sec / 5-10 sec / -	5-10 sec / 5-10 sec / **0.14-0.25** sec
Read CIM.eap	3 min / 29 sec / -	3 min / **20** sec / **7.5** sec
(with docgen.on=true: ~355 exported diagrams)	(+50 sec)	(+50 sec)
Read IEC61850.eap	2.8 min / 26 sec / -	2.8 min / **17** sec / **6.4** sec
(with docgen.on=true: ~200 exported diagrams)	(+32 sec)	(+32 sec)
Read base-small.eap	34 sec / 10 sec / -	34 sec / **8** sec / **1.3** sec
(with docgen.on=true: ~93 exported diagrams)	(+12 sec)	(+12 sec)

MS Word-dependent operations:

Execution times with jCleanCim-01v08

MS Word doc generation	01v07 (docgen.saveReopenEvery) duration	01v08 (docgen.saveReopenEvery) duration	speed improvement
IEC61970-301 Base (637 tab, 67 fig)	(12) 43.3 min	(27) 9.5 min	4.6 x
IEC61968-11 (378 tab, 37 fig)	(16) 12.1 min	(24) 4 min	3.2 x
IEC62325-301 (637 tab, 40 fig)	(16) 36.5 min	(24) 7.9 min	4.6 x
IEC61970-302 Dynamics (387 tab, 178 fig)	(12) 18.5 min	(27) 6 min	3.1 x
IEC61850-7-3 including a subset of IEC61850-7-2, with special table formatting (134 tab, 31 fig)	(12) 4.4* min	(5) 3.9 min	1.1 x*
IEC61850-7-4, with special table formatting (244 huge tab, 40 fig)	(12) 41.4 min	(27) 27.5 min	1.5 x
base-small (102 tab, 27 fig)	(12) 0.8 min	(5) 0.7 min	1.1 x**

Instead of conclusion

• • •

Known issues

Related to MS Word doc generation only – due to troubles with its automation API capability

- With some IEC templates, automatic update of tables of contents/tables/figures does not work properly
  - Workaround: After doc generation, do manual update of all TOCs
- Incorrect numbering of captions in the auto-generated document
  - Workaround: Before doc generation, first stop tracking changes, then do update of all TOCs. If you have deleted figure/table captions, ensure you accept those changes (because automation API returns them even when marked as deleted, as long as they are present in the template)!
- Word pop-up window "memory insufficient. Do you want to continue?“
  - Workaround: Before doc generation, in the input template, disable spell checking and change tracking
- Exceptions when using localised versions of MS Word, due to style names
  - Workaround: Install English language pack in MS Office
  - Planned to support non-English versions of MS Office in the next release

If acceptable by IEC, we may want to go simply with some XML one day...

Success stories

Generating official MS Word documents for CIM :

- IEC 61970-301 since Ed.4 (base CIM14)
- IEC 61968-11 since Ed.1 (DCIM10)
- IEC 62325-301 since CDV (market CIM01), and all EU profile documents
- IEC 61970-302 NWIP(Dynamics)
- Documentation of CIM extensions for various projects

Also:

- IEC 61850-7-4, IEC 61850-7-3 and IEC 61850-7-2 since their Ed.2.1
- Work ongoing for IEC 61850-7-410 (hydro), IEC 61850-7-420 (DER), IEC 61400-25-2 (wind) and IEC 61850-90-3 (condition monitoring)
- ***And IEC 61850-90-4 (comm. network engineering), the first official document from the IEC61850 family with the data model automatically generated from UML ☺***

Want more features?

jCleanCim is being developed on volunteering basis.

Consequently:

- The policy is to first support the immediate needs of official IEC TC57 UML models' editors (CIM and IEC61850 families), for their tasks of UML model management:
  - For the IEC process, and,
  - For the user community.
- This includes:
  - Bug fixes
  - New features that automate model managers' tasks (e.g. doc generation for CIM profiles, CIM profiles-UML cross-check)
  - New features that are easy to add (e.g., new validation rules, improved logging, better validation filtering and such)

If you want more advanced features (like GUI):

- Get involved – any help is welcome !

Credits

Authors of all the cool open source libraries and tools we use (see readme.html).

Lars-Ola Osterlund (ABB, Sweden) encouraged development of documentation generation for IEC 61850 UML model donated to IEC TC 57, to illustrate at least one benefit of using UML in WG10.

Kendall Demaree (Alstom, US), while being CIM model manager in 2008, developed open source CIMinEA application that helped us move standard CIM from Rational Rose to Enterprise Architect, and which allowed document generation for IEC 61970-301, Ed. 3. That work inspired some of the functionality related to document generation in jCleanCim.

Hubert Kirrmann (ABB, Switzerland), Laurent Guise (Schneider Electric, France) and other members of the WG10 61850 UML task force motivated development of the extended functionality for the needs of validation and document generation for IEC 61850 family of standards from UML being developed by the task force.

Pat Brown (EPRI, US) and Christoph Fleischer (ABB, Switzerland) have been continuously providing a valuable feedback as users of jCleanCim, which resulted into improved quality, and enhanced and new features.

Special thanks to Laurent Guise (Schneider Electric, France), who has been providing prototype implementation for a couple of new features and issue fixes – most of which have been ported to the jCleanCim architecture.

structure101

Headway software, which provided open source license for their great tool Structure 101 (http://structure101.com/), used to keep the jCleanCim architecture clean.